WL-TR-96-3087

SCALEABLE EXPERT SYSTEMS FOR ADDING CRISP KNOWLEDGE TO PILOT-VEHICLE INTERFACES

PETER G. RAETH, ANTHONY J. MONTECALVO, JAMES L. NOYES

MAY 1996

FINAL REPORT FOR 01/01/93-03/30/96

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

FLIGHT DYNAMICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT PATTERSON AFB OH 45433-7562

NOTICE

This technical report has been reviewed and is approved for publication.

Peter G Raeth

PETER G. RAETH, MAJOR, USAF
CHIEF, PILOT-VEHICLE INTERFACE TECHNOLOGY SECTION
WRIGHT LABORATORY

John Reising

JOHN M. REISING, PHD
ACTING CHIEF, PILOT-VEHICLE INTERFACE BRANCH
WRIGHT LABORATORY

David K. Bowser

DAVID K. BOWSER
ACTING CHIEF, FLIGHT CONTROL DIVISION
WRIGHT LABORATORY

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 1996

3. REPORT TYPE AND DATES COVERED
Final 01/01/93 - 03/30/96

4. TITLE AND SUBTITLE
Scaleable Expert Systems for Adding Crisp Knowledge to Pilot/Vehicle Interfaces

5. FUNDING NUMBERS
PE 62201
PR 2402
TA 04
WU 86

6. AUTHOR(S)
Peter G. Raeth
Anthony J. Montecalvo
James L. Noyes

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Flight Dynamics Directorate; WL/FIGP
Wright Laboratory
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7521

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Flight Dynamics Directorate; WL/FIGP
Wright Laboratory
Air Force Materiel Command
Wright-Patterson AFB OH 45433-7521

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
WL-TR-96-3087

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Vehicle systems operations, from automobiles, to airliners, through craft for sea and space, are heavily dependent on automated uses of data generated by both on-board and off-board sources. Not only are vehicles affected, but so are diverse systems such as process, manufacturing, and power generation plants. Historically, the volume of data in each instance keeps growing because of such factors as task and system complexity. Other factors are environmental, accuracy, safety, cost-effectiveness, and customization requirements. A crowded operational domain is also a factor for vehicle operations and communications networks. Tragedies such as that which occurred at the Three-Mile Island nuclear power plant and during the V-22 Osprey Tiltrotor crash are examples that point out the volume and complexity of the data and automation involved. These incidents also point out that it is essential to have a scaleable framework for the organization and prioritization of raw data. In this way, the operator gets pertinent information necessary for reaching critical decisions. This framework also should support decision aids that offload operational details so that human resources can be applied more appropriately to goal achievement and other management issues. This paper outlines such a framework and gives examples.

14. SUBJECT TERMS expert systems, knowledge-based systems, rule-based systems, artificial intelligence, real-time processing, parallel processing, distributed computing, decision support, pilot/vehicle interface, human/systems interface, crew/systems interface

15. NUMBER OF PAGES
92

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
unclassified

20. LIMITATION OF ABSTRACT
UL

TABLE OF CONTENTS

Appendix:

LIST OF FIGURES

ACKNOWLEDGEMENTS

The authors are indebted to Dr John Reising of Wright Laboratory for his helpful comments throughout this project. We also appreciate the administrative support of Ms Amy Graves and Ms Jene Peebles.

Parts of this research were sponsored by the Air Force Office of Scientific Research as part of the summer faculty program.

1.0 INTRODUCTION

Modern processes and systems, both fixed and mobile, use increasingly large volumes of situational data. They also require faster response rates. This reality pushes human operators to their limits of cognition and memory, causing systems to be difficult to operate for a single human without automated assistance. Thus, humans become supervisors rather than operators (see Sheridan, Moray and Lee, Pawlowski and Mitchell, and Wickens).

The processes and systems in question are found in such diverse domains as manufacturing, power generation, health monitoring, financial analysis, and vehicle operations. They can become difficult to operate because of their speed, complexity, and the huge amounts of data involved. The data rates keep growing because of such factors as task, process, and system complexity. Other factors are environmental, accuracy, safety, cost-effectiveness, and customization requirements. A crowded operational domain is also a factor for vehicle operations and communications networks. Large volumes of raw data must be collected and integrated in real-time to determine the existence of system conditions. These conditions lead to decision criteria and ultimately to action. Expert systems technology is the basis of a scalable framework that can do this effectively.

It is important to understand this paper's definition of "system" and "real-time." "System" refers to the ambient environment, the operator, and devices that may or may not be under the operator's control or under the control of automation directly supporting the operator. "Real-time" refers to a response that occurs fast enough to have an impact on the current situation. Both human and automation have to operate in "real-time." The human's ability to do this without automated assistance is sometimes in question, as illustrated by Scott's statement:

> Despite the extensive goof-proof designs incorporated into the F-15E, the simplest of procedures can and will be forgotten in the heat of battle. Some pilots privately question whether the increasing sophistication of aircraft such as the F-15E is starting to surpass a human being's capability to deal with complexity that takes a high degree of conscious thought.

Thus, one often hears of the human/electronic crew. The periodic Joint German/British/USA Workshop on Human/Electronic Crew Teamwork sponsored by the European Office of Aerospace Research and Development (see Emerson and Reising) discusses this topic. Figure 1 shows that it is essential to solve the human/electronic crew teamwork puzzle. As this figure illustrates, task overloading prevents a human-only system from reaching its target operational capability.

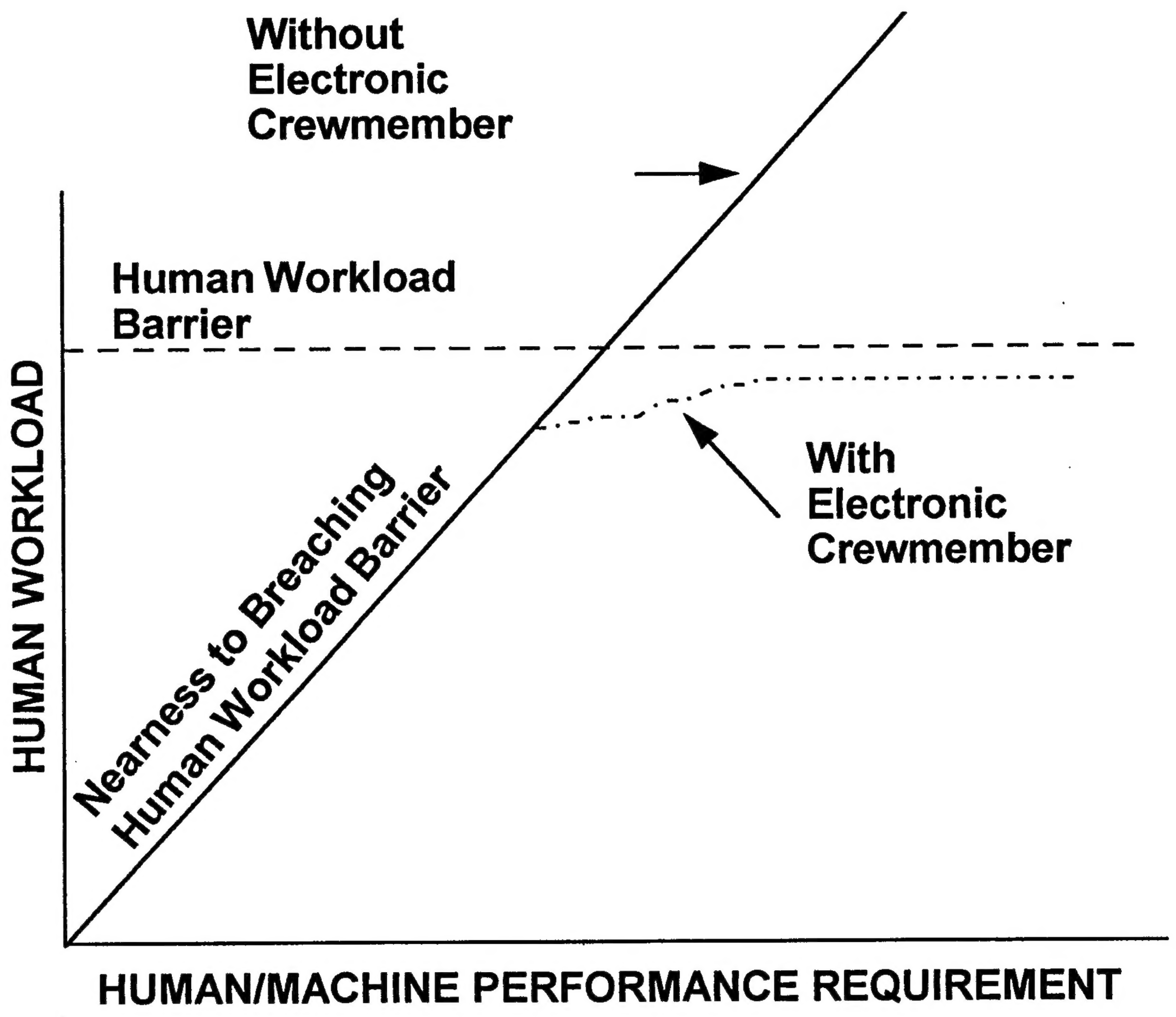


Figure 1: Overcoming Task Overload

1.1 BASIC EXPERT SYSTEMS FORMULATION

The value of expert systems to help solve a variety of diagnostic and advisory needs has been well-demonstrated over the last two decades, as discussed by Noyes(93). Sometimes, a large number, say 1000, "rules" must be continuously checked in real-time (e.g., time steps of every 0.1 seconds) due to stringent requirements imposed by the problem. In addition, while each rule may use only around 10 criteria, there may be a very large number of possible criteria, say 100000, in the knowledge base. These must be checked during each time-step. Because of these timing demands, the expert system must scale so that the timing demands can be met in spite of expanded knowledge volume. This paper presents expert systems techniques that scale in a natural way for parallel processing architectures. Parallel processing has become increasingly important in order to accelerate a variety of computations, as discussed by Noyes and Trippi & Turban.

Techniques for two expert systems will be developed:

1) Aircraft Condition Analysis: observes raw data gathered over time from various sources to determine specific conditions evident in the aircraft, crew, and the surrounding environment.

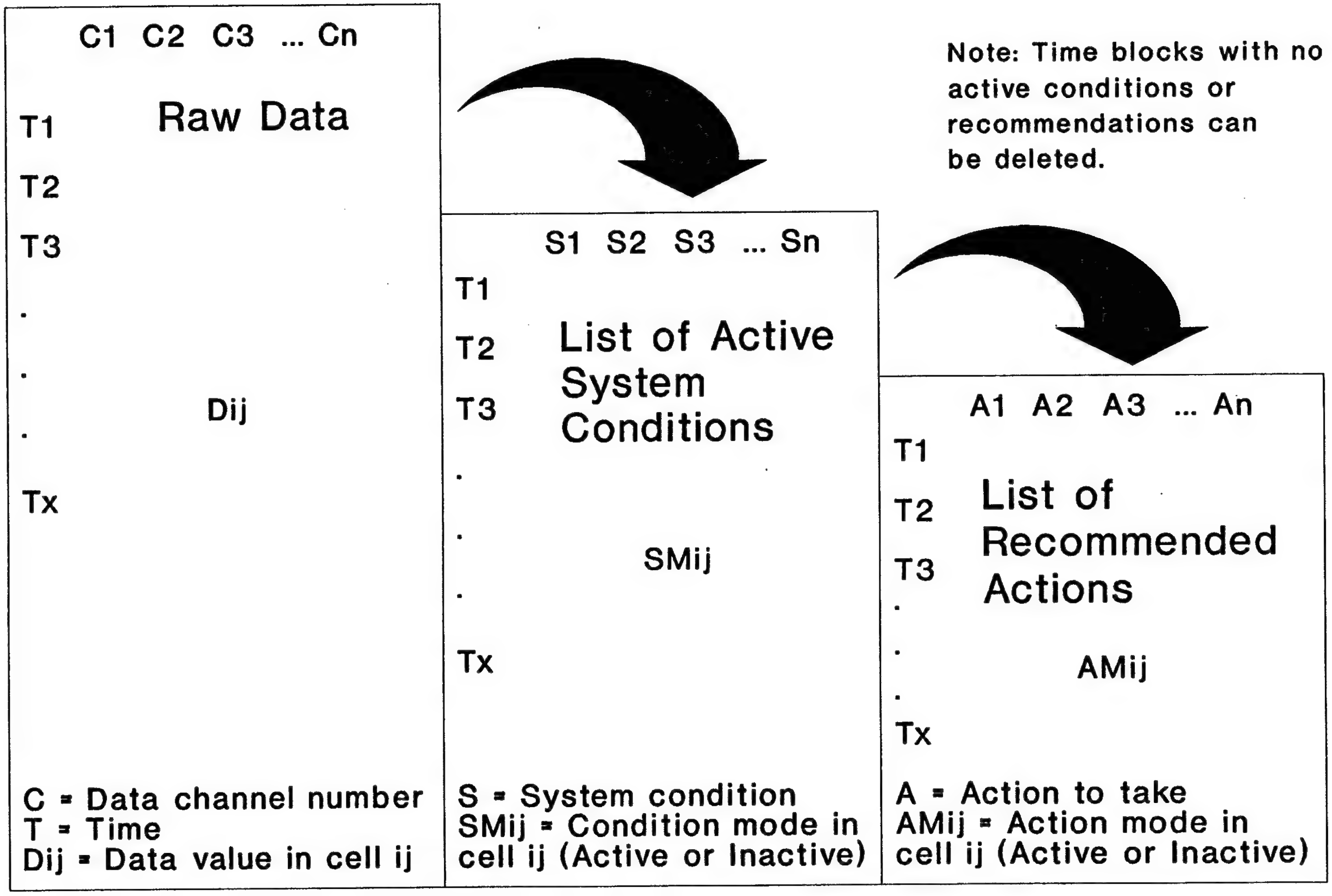
2) Action Determination: observes the conditions evident in the aircraft, crew and surrounding environment to select actions to take. The conditions are decision criteria that, when satisfied, lead to action.

Used together, these techniques provide a scaleable framework that organizes raw data so that existing systems conditions may be determined and appropriate action recommended or taken. The human/electronic crew operates with two major sets of decision criteria. The first set of criteria allocates action to the computer or the human. The second set determines when a given action is appropriate. The criteria are derived from the system conditions evident in the raw data. Raw data must be collected and integrated to determine system conditions that lead to decision criteria and ultimately to action. Expert systems technology provides a way to integrate and use high-speed raw data effectively to achieve this end.

The first technique to be discussed detects system conditions implied by raw data taken from various on-board and off-board sources. The second technique uses these conditions as decision criteria to generate appropriate actions, recommendations, and information displays. Domain experts use simple wordprocessors to generate the knowledge bases supporting these techniques. The result is automatically translated into appropriate discrimination networks and matrices for use by the inference engines. The techniques permit non-exhaustive search which still yields complete results based on available facts. They avoid the use of special syntax and terminology. Further, they allow knowledge base maintenance by a domain expert instead of requiring expert systems or programmer specialists.

Modern systems typically contain a host of raw data from which its condition may be determined. This information on conditions can be used to determine appropriate actions. Figure 2 illustrates the flow of raw data from system conditions to recommendations and action. As the data move through this scenario, they become lower in volume but higher in value relative to the goals we have in observing the system.

Figure 2 illustrates three transformation phases: raw data collection, condition analysis, and action determination. The computer must pass the raw data through condition analysis to recommend appropriate action using the knowledge supplied by the domain expert. Each of the three phases will be discussed in this paper, along expert systems implementation details. Besides the information in this paper, related discussion may be found in the works by Raeth, Noyes, and Montecalvo (see the bibliography).



C1 C2 C3 ... Cn

T1 Raw Data

T2

T3

.

.

Dij

.

Tx

C = Data channel number
T = Time
Dij = Data value in cell ij

S1 S2 S3 ... Sn

T1

T2 List of Active System Conditions

T3

.

.

SMij

.

Tx

S = System condition
SMij = Condition mode in cell ij (Active or Inactive)

A1 A2 A3 ... An

T1

T2 List of Recommended Actions

T3

.

.

AMij

.

Tx

A = Action to take
AMij = Action mode in cell ij (Active or Inactive)

Note: Time blocks with no active conditions or recommendations can be deleted.

Figure 2: Transforming Raw Data Into Relevant Information

1.2 DEFINITIONS

domain expert - the individual or team that creates the knowledge base used to drive an expert system, all other functions (including inferencing and memory structuring) are performed automatically

expert system - the software, knowledge bases, data source interfaces and user interfaces used to perform decision support functions

inference engine - that portion of the expert system software that determine truth values and resulting actions by traversing the memory structure created from the knowledge base

inferencing - the act of exercising the inference engine

knowledge base - information generated by the domain expert to drive the expert system, this information is in the language of the domain expert and uses no special syntax or terminology

knowledge base evaluation - same as inferencing

1.3 AN EXAMPLE APPLICATION

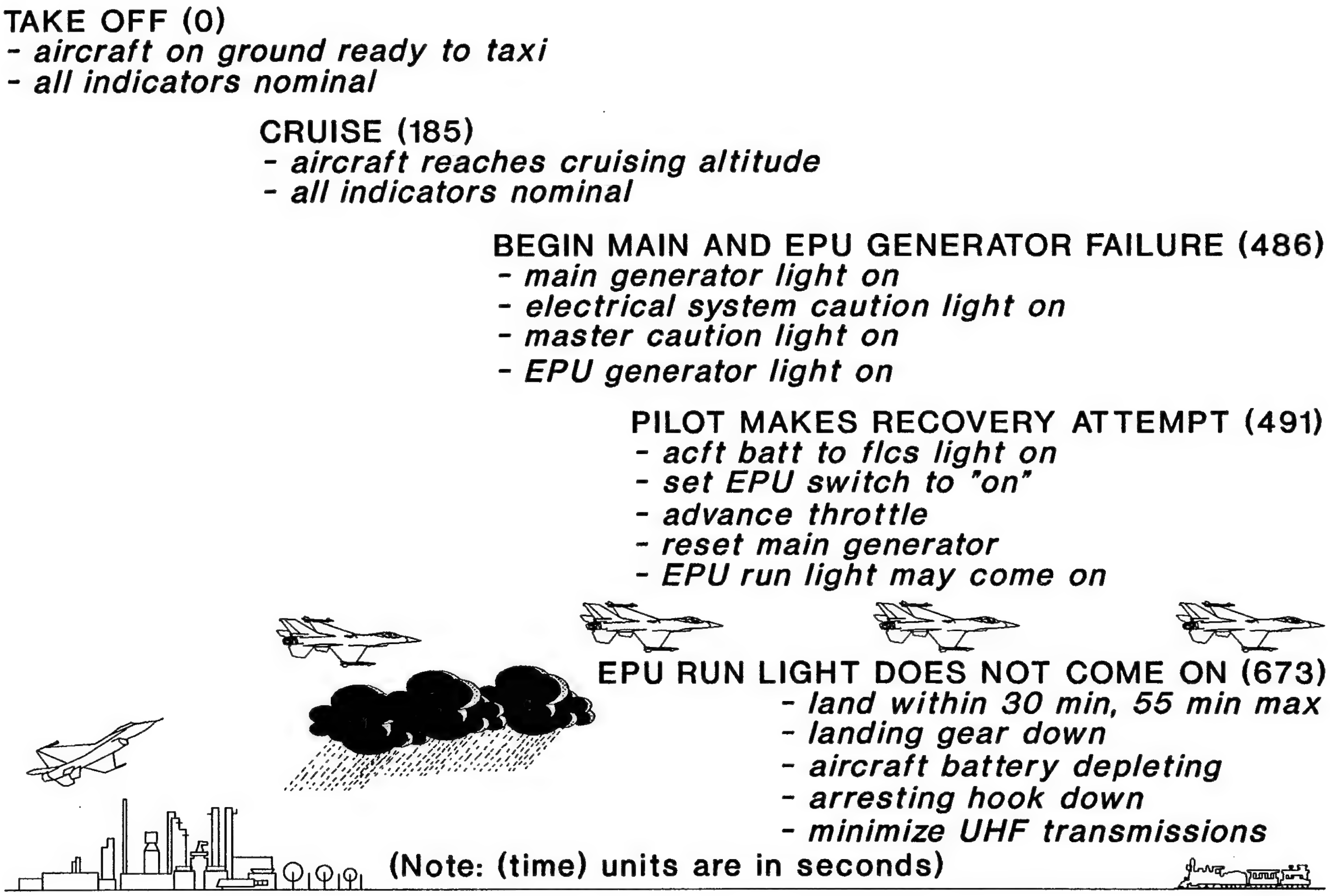
As we go into detail on how the basic formulation described above might be implemented, it will be easier to explain if we use a concrete example. This example will involve aircraft electrical generator failures.

Electrical generator failures in aircraft are a serious consideration in flight safety. While flight may be maintained for a given time period by using battery power, a landing must soon be conducted unless the generators can be brought back on line. Sufficient time usually exists to maintain flight safety but, as Harvey observed after the V-22 Osprey Tiltrotor crash, the operator must have sufficient information during that time to recover from the failure. If the operator cannot cope with the situation, the aircraft's computers have to have sufficient tasking capability to take charge.

This paper selects simplified features of an aircraft generator failure to illustrate data fusion, information fusion, and decision support via expert systems. For an exact description of aircraft failure modes, the reader should select the operator's guidance documents for a specific aircraft.

Briefly put, when the main and backup generators on an aircraft fail, electrical systems automatically switch to battery power. At that point, the battery begins to deplete. If the generators cannot be brought back on line, as little as one hour of flight time remains, for some aircraft. After that, the flight control system becomes so erratic that a crash is inevitable. Figure 3 illustrates a typical scenario.

This failure example is an excellent means for illustrating the expert systems techniques for turning raw data into system conditions information. From there, this information is used to generate recommendations and actions.



TAKE OFF (0)
- *aircraft on ground ready to taxi*
- *all indicators nominal*

CRUISE (185)
- *aircraft reaches cruising altitude*
- *all indicators nominal*

BEGIN MAIN AND EPU GENERATOR FAILURE (486)
- *main generator light on*
- *electrical system caution light on*
- *master caution light on*
- *EPU generator light on*

PILOT MAKES RECOVERY ATTEMPT (491)
- *acft batt to flcs light on*
- *set EPU switch to "on"*
- *advance throttle*
- *reset main generator*
- *EPU run light may come on*

EPU RUN LIGHT DOES NOT COME ON (673)
- *land within 30 min, 55 min max*
- *landing gear down*
- *aircraft battery depleting*
- *arresting hook down*
- *minimize UHF transmissions*

(Note: (time) units are in seconds)

Figure 3: Electrical Generator Failure Scenario

2.0 COLLECTING AND ORGANIZING RAW DATA

The collection and organization of raw data are crucial steps in the observation of any system. Data from each source must be gathered, recorded, converted, and passed on to the condition analysis phase. Figure 4 shows these basic steps. In this paper we will not deal with the physical implementation of data acquisition since that is a separate topic.

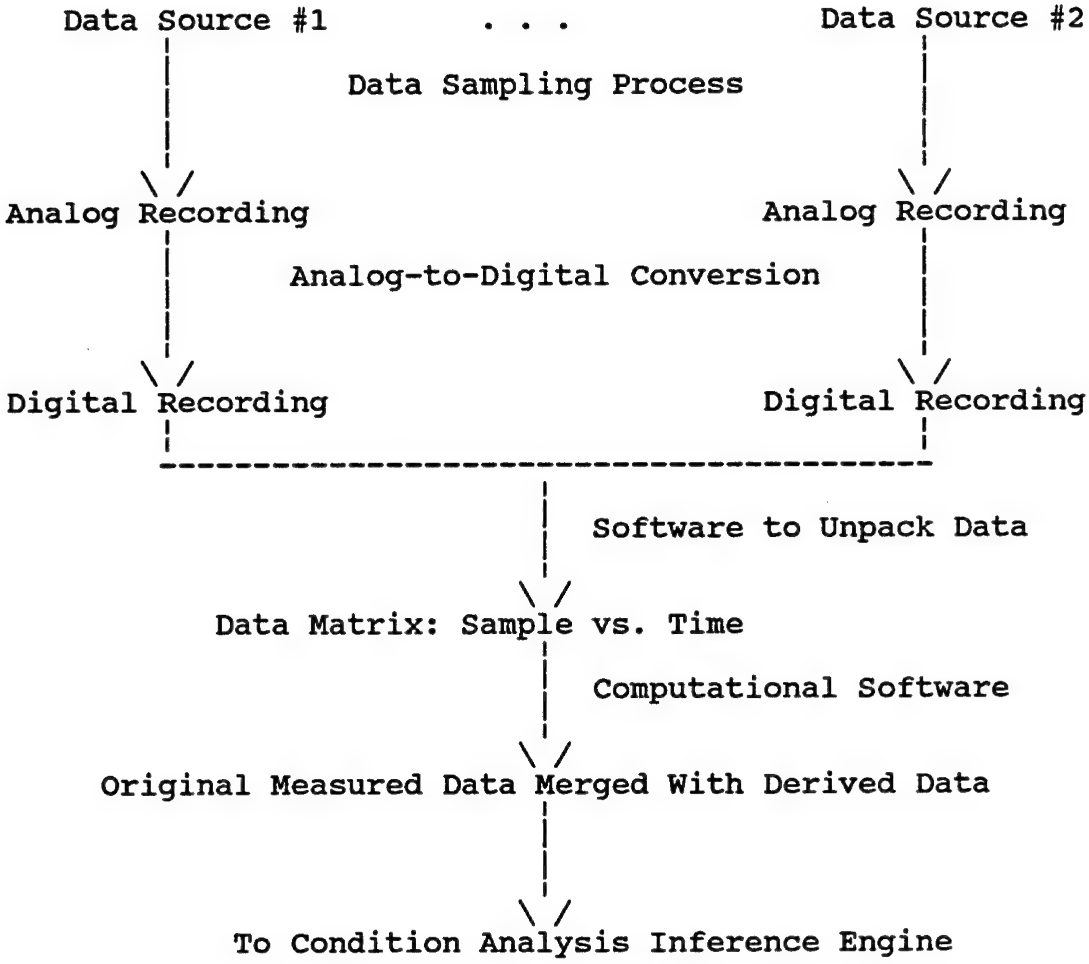


Figure 4. Collection and organization of raw data for the condition analysis inference engine

According to Figure 4, there are two types of raw data needed by the condition analysis inference engine, measured and derived. Measured data come directly from system sampling. Derived data are the result of computations performed on measured data. Examples of derived data are Fourier transforms, averages, slopes, standard deviations, and other results that can be directly computed using measured data. Derived data are merged with measured data to create additional data matrix columns. In a data matrix, the measured and derived data are the columns and the time stamps are the rows. Lets see how this relates to the generator example.

In a typical advanced jet fighter, there are main and backup electrical generators. These generators power critical systems, such as flight control. If these generators both fail, a battery backup provides electrical power for about an hour. During that hour, the aircraft must land or suffer the result of an erratic flight control system. Power to other subsystems critical to flight safety is lost too. A crash is inevitable if the aircraft is not landed within 1 hour after going on battery power.

There are several causes of main and backup generator failures. Some may be remedied by the pilot. Once the cockpit indications of failure occur, the decision support system can lead the pilot through the remedy procedure relative to the causes of failure. If the pilot is too busy with the flight mission, the remedy procedure can be performed automatically.

The remedy procedure for a particular instance of failure typically relies on a constant knowledge of the condition of the aircraft. The aircraft's condition is determined by observing the raw data available over time in the cockpit. For the main and backup generator problem, the following data have to be available to the automation:

Data Source #	Name	Units
1	system clock	seconds
2	master caution light	volts
3	main generator light	volts
4	main generator switch position	select
5	epu generator light	volts
6	epu generator switch position	select
7	electrical system caution light	volts
8	epu run light	volts
9	battery depleting light	volts
10	acft batt to flcs light	volts
11	altitude	feet
12	air speed	knots
13	landing gear position	degrees
14	arresting hook position	degrees
15	air speed slope	knots/sec

acft:	aircraft
batt:	battery
epu:	emergency power unit
flcs:	flight control system
gen:	generator
pwr:	power

The operator's guide to a particular aircraft may suggest many other important parameters. The parameters listed above were chosen to provide a meaningful example. The raw data from the sources that gather these parameters are passed on to the expert system in time sequence via time-stamped records. Figure 5 shows a partial set of example records.

TIME	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	5	0	3	0	0	0	0	0	0	-90	0	1
1	0	0	5	0	3	0	0	0	0	0	10	-90	0	10
2	0	0	5	0	3	0	0	0	0	0	20	-90	0	10
3	0	0	5	0	3	0	0	0	0	0	30	-90	0	10
4	0	0	5	0	3	0	0	0	0	0	40	-90	0	10
5	0	0	5	0	3	0	0	0	0	0	50	-90	0	10
6	0	0	5	0	3	0	0	0	0	0	60	-90	0	10
7	0	0	5	0	3	0	0	0	0	0	70	-90	0	10
8	0	0	5	0	3	0	0	0	0	0	80	-90	0	10
9	0	0	5	0	3	0	0	0	0	0	90	-90	0	10
10	0	0	5	0	3	0	0	0	0	0	100	-90	0	10
11	0	0	5	0	3	0	0	0	0	0	110	-90	0	10
12	0	0	5	0	3	0	0	0	0	0	120	-90	0	10
13	0	0	5	0	3	0	0	0	0	0	130	-90	0	10
14	0	0	5	0	3	0	0	0	0	0	140	-90	0	10

Figure 5. An example of a time-sequenced data records
(Top row is data source #)

Each record of Figure 5 has columns that represent the readings taken from each data source at a given moment in time. The column for Data Source #1 contains a reading from a clock that is counting the number of time units since the start of flight activity. The other columns contain the value taken (sampled) from the other sources at a given system clock time. The units of any particular column are of no consequence to the expert system itself. The expert system will function correctly as long as the units of each column are consistent with the units used by the domain expert who created the knowledge base. If the domain expert entered numbers thinking "volts," for instance, then no readings should be taken in millivolts. By the time any data readings or knowledge base information reaches the inference engine, measurement units are no longer a consideration.

3.0 INTERPRETING RAW DATA AS SYSTEM CONDITIONS

Raw sampled data are, by themselves, much less useful than even mechanical gauges. A mechanical gauge gives the observer a direct indication of present value, value trend, and rate of change. These indications are tremendously more valuable than any single data record. By putting expert systems to work it is possible to make raw sampled data take on a whole new flavor.

The raw data, taken over time, imply that the aircraft is in a certain condition. Example aircraft conditions are "master caution light on" and "angle of attack below minimum required." Other examples are "main or backup generator light blinking" and "battery depleting." The pilot/vehicle interface designer needs a way to collect what is known about raw data and how it is used to indicate aircraft conditions that become decision criteria supporting recommendations or actions.

Two things must happen to support this aircraft condition analysis phase. First, the knowledge of the aircraft expert must be captured. Second, this knowledge must be used consistently, with the raw data, to determine when aircraft conditions are active.

The supporting mechanism for knowledge capture helps the aircraft expert gather and organize what is known into a detailed description of how conditions are recognized in raw data. Based on extensive interviews with experienced aircraft and pilot/vehicle experts, three basic categories of information make up this knowledge base:

CONDITIONS: internal or external aspects of the aircraft's situation that remain active for a given time period.

EVENTS: correlations of raw data that must be observed in time sequence before a condition is determined to be active.

SPECIFICATION SETS: triplets that identify a data source, its sampled value, and a numerical comparison operator (<, >, =, <=, =>, and <>). A list of these sets makes up an event. The sets in an event must all be recognized simultaneously before the event is determined to have occurred. Specification sets are used to identify events, and events are used to identify conditions. The comparison operators are integrity constraint ranges. Integrity constraints are restrictions on the real world for which a model is valid, according to Gal and Minker.

Besides the three basic categories mentioned above, there are two categories of related information that also must be collected:

DWELL TIME: the minimum length of time an event must last.

TRANSITION TIME: the maximum length of time between the start of one event and the beginning of the next event.

It is very important to devise a mechanism oriented to the aircraft expert for the capture of these five categories of information. The mechanism chosen must use the language of the expert and be simple enough for that person to modify the knowledge base incrementally. The authors' personal experience is that a common spreadsheet does nicely when set up as illustrated in Figure 6. Once the spreadsheet is filled in, one can "print" the appropriate rows and columns to disk as a text file for later use by the inference engine's preprocessor. This "spreadsheet" can be set up on a wordprocessor.

		EVENT TIMING		EVENT SPECIFICATION SETS					
CONDITION NAME	\	DWELL	TRANSITION	S #	VALUE	COP	S #	VALUE	COP
main gen light		1	0	3	3	>=	3	6	<=
main gen light	\	2	0	3	3	<			
epu generator switch on		1	0	6	5	>=			
epu generator switch on	\	1	0	6	5	<			
epu gen light still on		180	0	5	3	>=	5	6	<=
epu gen light still on	\	2	0	5	3	<			
main gen light still on		180	0	3	3	>=	3	6	<=
main gen light still on	\	2	0	3	3	<			
single throttle advance		5	0	3	3	>=	3	6	<=
		5	0	5	3	>=	5	6	<=
		5	0	8	3	<			
		5	0	6	5	>=			
single throttle advance	\	5	0	8	3	>=	8	6	<=
master caution light		1	0	2	3	>=	2	6	<=
master caution light	\	2	0	2	3	<			
acft batt to flcs light		1	0	10	3	>=	10	6	<=
acft batt to flcs light	\	2	0	10	3	<			
acft batt depleting light		1	0	9	3	>=	9	6	<=
acft batt depleting light	\	2	0	9	3	<			
batt depleting 5 min		1	0	9	3	>=	9	6	<=
		299	0	1	0	>			
batt depleting 5 min	\	2	0	9	3	<			
batt depleting 5 min	\	300	0	1	0	>			
landing gear down		2	0	13	-90	<=			
landing gear down	\	2	0	13	-90	>			
arresting hook down		2	0	14	-90	<=			
arresting hook down	\	2	0	14	-90	>			
elec sys caution light		1	0	7	3	>=	7	6	<=
elec sys caution light	\	2	0	7	3	<			
landing required (elec)		1	0	7	3	>=	7	6	<=
landing required (elec)	\	2	0	13	0	<=			
		2	0	15	0	=			
		2	0	11	0	<=			
airspeed up for 10 sec		10	0	15	0	>			
airspeed up for 10 sec	\	2	0	15	0	<=			

S #: Data Source # VALUE: Expected Value
COP: Comparison Operator
\: Anti-Condition (how to tell when the named condition is no longer active)

Figure 6. Example knowledge capture spreadsheet for the aircraft condition analysis inference engine

What kind of information might be collected in this spreadsheet? One could start by setting maximum and minimum values for the samples to be taken from each data source. These values define broad domain limits. For instance, for a given condition, the angle of attack may be too high or too low. That fact could be noted and a suitable recommendation given or action taken. One flight safety consideration would be too low an altitude without being in a landing or other low-level flight configuration. This could well indicate a need to raise altitude immediately, especially if altitude is still decreasing. These are just some basic ideas. For completeness, a rigorous analysis of the aircraft and its mission requirements should be accomplished and a suitable knowledge base generated.

The inference engine's preprocessor transforms the spreadsheet shown in Figure 6 into a discrimination network. The format of the spreadsheet lends itself to this transformation. Notice that each row of the spreadsheet represents an event with that event's specification sets. The condition named at the beginning of the row is associated with the the event on that row. If there is no condition name and the line is not totally blank, then the event on that row is the next sequential event for the previously named condition. Separate linked nodes are created in the computer's memory to associate each condition with its events and each event with its specification sets. Figure 7 gives a formal presentation of the basic procedure.

1. Open the file containing the spreadsheet. Create a null discrimination network. Skip all leading records starting with blank or '*' characters (these are comment lines).

2. Read the next sequential record from the file. Skip all records containing only blanks or starting with '*'. If end-of-file is encountered then the network is completed.

3. If the non-empty record's first character is a blank then link a new event node to the end of the current condition's event list. Otherwise, link a new condition node to the condition list and link the first event node to that condition.

4. Get the dwell and transition times from the record and store them in the current event's node. Get the first specification set from the record and link the first specification set node to the current event.

5. Store the data source #, value, and comparison operator in the current specification set node.

6. Get the next specification set from the current record. If there are no more specification sets, return to Step 2.

7. Create a new specification set node, link it to the previous specification set node, and return to Step 5.

Figure 7. Basic procedure for building a discrimination network

Figure 8 shows the discrimination network developed from Figure 6. Physically, this network is implemented as a series of connected linked lists. The conditions are listed vertically, the events for each condition are listed horizontally, and the specification sets for each event are listed diagonally. There is no logical limit to the number of conditions, events, or specification sets. The only physical limitations are the amount of time and memory available to the system. Each condition is evaluated independently. Therefore, this framework should scale well across multiple processors as more conditions are added to the knowledge base.

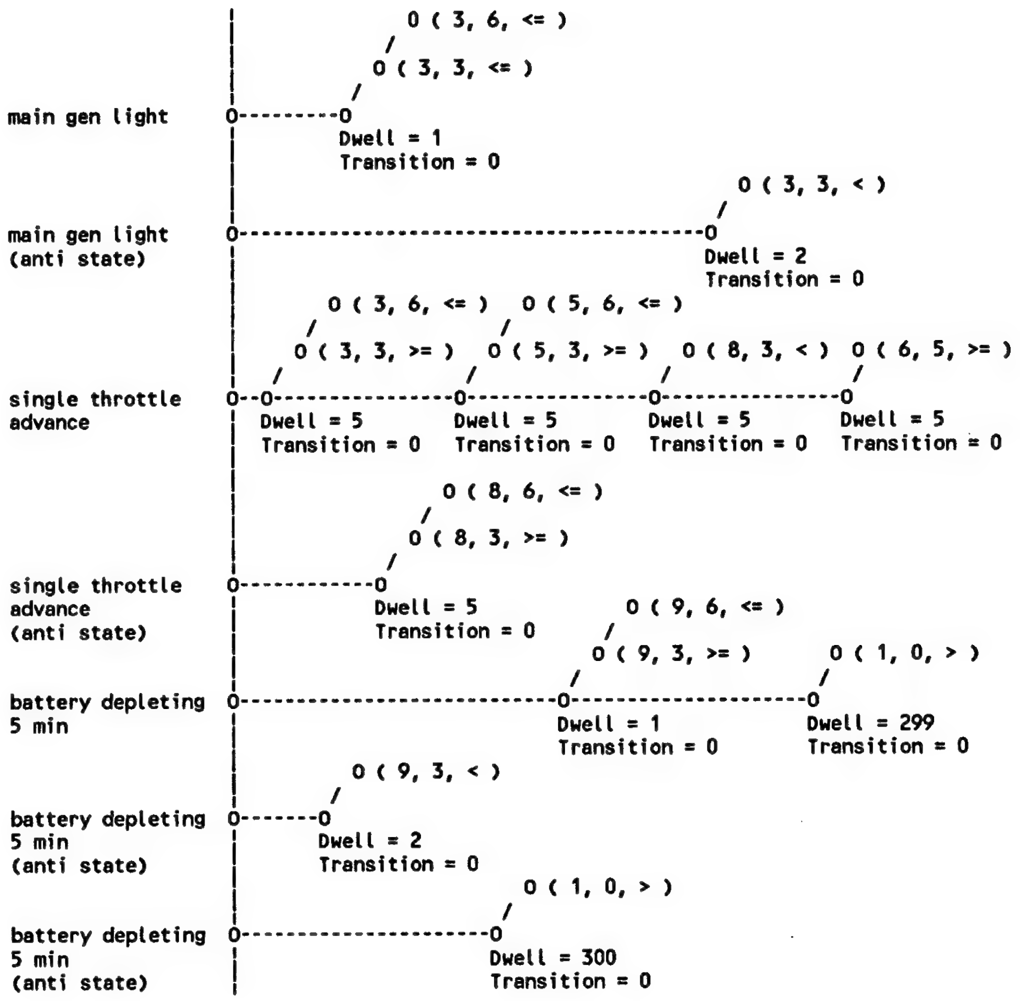


Figure 8. Discrimination network based on example condition analysis knowledge base
(O = a data network node stored in memory)

The network shown in Figure 8 is traversed in a forward-chaining, depth-first fashion (top-down, left-right). The inference engine uses the network to get the information it needs to tell when given specification sets, events, and conditions occur. The network is traversed while each data sample is compared to the specification sets of the current event of each condition on the active search list. Once a specification set is found that does not compare as it should, the comparison process for that condition stops and the comparisons for the next condition start. If all the specification sets compare correctly and their event's dwell and transition time factors are within range, the current event is passed and the next event is made the current event for comparison purposes on the next sample. Once all events for a particular condition are passed, that condition is identified and placed on the active condition list. Active conditions remain so until one of their anti-conditions is found. If an active condition lacks an anti-condition, that condition is reported when it occurs and then taken off the active list. A formal presentation of the basic procedure is given in Figure 9.

1. Initialize the discrimination network
    a. Set the condition pointer to the top of the network.
    b. Set the current event pointer for each condition to the first event in that condition's event list
    c. Reset the dwell and transition time counters for all events of all conditions
    d. Read the first data sample

2. For the current event of the current condition compare the event's specification sets to the sampled data.

3. If all specification sets for the current event compare correctly to the sampled data then increment the dwell time counter for that event. Otherwise, the event is not found and the dwell time counter is reset. In any case, increment the transition time counter.

4a. If the dwell time for the current event is satisfied then reset the dwell and transition time counters for this event and set the current condition's current event pointer to the next event in the list. If there is no other event then reset this condition's current event pointer to the first event in the event list. This condition is now considered active until its anti-condition is found.

4b. If the transition time for the current event is exceeded without the event having been found then reset its transition and dwell time counters and set the current event pointer for the current condition to the first event in the condition's list.

5. Set the network's condition pointer to the next condition in the network. If there is no other condition then point to the first condition in the network and read the next data sample. In any case, except end-of-data, return to Step 2.

Figure 9. Basic sequence of actions in the traversal of the condition analysis discrimination network

3.1 AN EXAMPLE TIME PERIOD

Now that we have looked at the details of the condition analysis expert system, let's see how it would perform with some sampled data.

The knowledge base in Figure 6 asks the inference engine to look for several conditions in the sampled data:

1) MAIN & EPU GEN FAILURE: When the main or EPU generators fail to produce power for the electrical systems, a special battery-driven trigger is raised to the 6-volt level. When the generators are operating properly, this trigger is below 3 volts. Each generator has its own failure trigger. A special procedure is followed when both generators fail at the same time.

2) MASTER CAUTION LIGHT: Most cockpits are designed such that any time a failure trigger is set, a master caution trigger is also set. This ultimately causes a general alert to the pilot that something is amiss and action is being recommended or taken.

3) BATTERY DEPLETION: As the battery power depletes during generator failure, a general loss of electrical power to the avionics occurs and the flight control system becomes increasingly erratic. A cockpit warning light indicates this condition prior to its affect on aircraft systems.

4) ARRESTING HOOK DOWN: When the pilot puts the arresting hook in the "down" position, the aircraft is in landing mode. At this point, it is expected that the pilot is trying to land the aircraft. Sensors read this position as -90 degrees.

5) PROGRESSIONS OF TIME: Existing conditions are flagged as having existed for a period of time. As time progresses, the pilot is reminded of actions needing to be taken, if they have not already.

Figure 10 shows an example of these conditions found by the inference engine in raw data. Figure 3 pictures the scenario for this example. The basic idea is that if the pilot fails to react to traditional cockpit cautionary signals, the automation will begin to react, giving additional warning and suggestions. In the long run, this reaction could become a dynamic allocation of the remedy procedure from pilot to computer. The time over which this example takes place is compressed in order to save space.

```
ANALYZING FILE GROUP f-16 AT TIME =     0
landing gear down BEGINS AT     0
airspeed up for 10 sec BEGINS AT     0
airspeed up for 10 sec EXISTS BETWEEN   0 AND  19, TIME LENGTH = 19
landing gear down EXISTS BETWEEN   0 AND  20, TIME LENGTH =  20
main gen light BEGINS AT   485
elec sys caution light BEGINS AT   487
landing required (elec) BEGINS AT   487
master caution light BEGINS AT   489
epu gen light BEGINS AT   495
epu generator switch on BEGINS AT   500
single throttle advance BEGINS AT   493
airspeed up for 10 sec BEGINS AT   511
airspeed up for 10 sec EXISTS BETWEEN 511 AND 563, TIME LENGTH = 52
main gen light still on BEGINS AT   485
acft batt depleting light BEGINS AT   673
epu gen light still on BEGINS AT   495
batt depleting 5 min BEGINS AT   673
arresting hook down BEGINS AT   678
batt depleting 5 min EXISTS BETWEEN 673 AND 974, TIME LENGTH = 301
batt depleting 10 min BEGINS AT   673
FINISHED AT TIME =  1300
```

Figure 10. Example raw data implying certain aircraft conditions have occurred

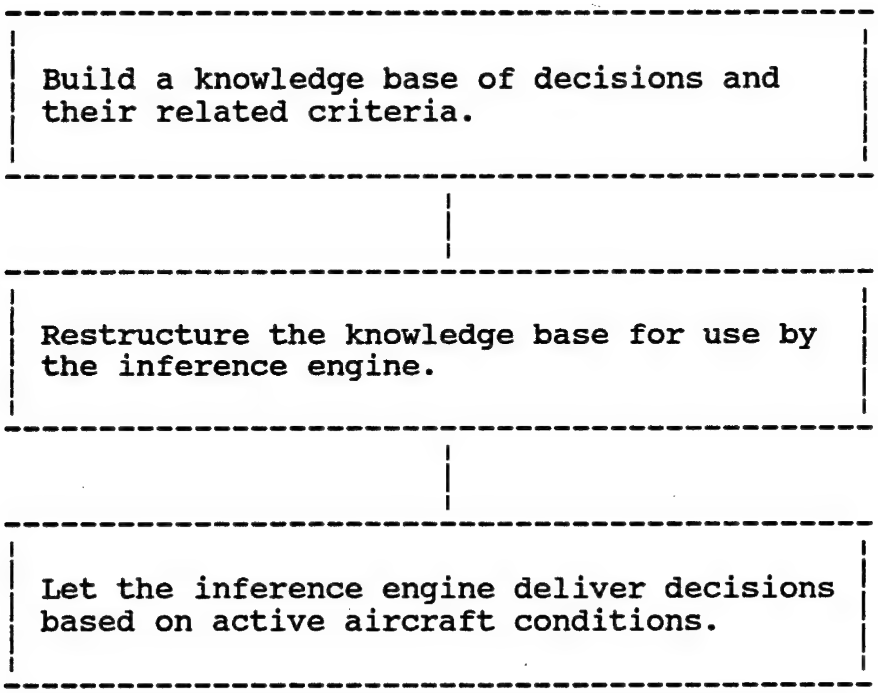
4.0 USING AIRCRAFT CONDITIONS AS DECISION CRITERIA

So far, we have seen how expert systems can be used to find the system conditions implied by raw data. An example based on an electrical generator failure was used to illustrate the integration of data from several sources to derive system conditions that lead to recommendations and action. We have also seen that it is not necessary to use special syntax for entering information into the knowledge base. Filling in the rows and columns of a simple spreadsheet will do as long as the elements are oriented to the domain expert. The resulting frames are automatically translated by the inference engine's preprocessor into a discrimination network.

We now continue with our aircraft example to show how expert systems can use active conditions as criteria to support recommendations and actions.

Aircraft conditions being active over a length of time can be an indication that certain recommendations or actions are appropriate. Recommendations and actions are the result of decisions supported by observations of active aircraft conditions that are implied by raw sensor samples. A recommendation may be something like "Raise altitude above 1500 feet" or "Increase throttle to extinguish hydrazine light." An action would carry out a recommendation.

To make use of what is known about aircraft conditions (criteria) and appropriate recommendations and actions (decisions), we need a way to build and use an expert system that associates decisions and their criteria. There is a three-step method that can be used to put together such an expert system. These steps are illustrated in Figure 11.



Build a knowledge base of decisions and their related criteria.

Restructure the knowledge base for use by the inference engine.

Let the inference engine deliver decisions based on active aircraft conditions.

Figure 11. The three basic steps in building an expert system for delivering decisions based on the use of active aircraft conditions as decision criteria.

Implementing the three basic steps shown in Figure 11 puts expert systems techniques to good use. Each of these steps are discussed in the following paragraphs.

4.1 BUILDING A KNOWLEDGE BASE OF DECISIONS AND THEIR CRITERIA

The act of building a knowledge base about decisions and their related criteria is done without regard to its later use by the inference engine. Thus, aircraft and pilot/vehicle interface experts can concentrate on the system in question rather than on computers, special syntax, or terminology unique to exper systems.

The bounding of the knowledge domain and the acquisition of knowledge are two classical bottlenecks in the development of expert systems. The knowledge acquisition method discussed here is designed to deal with these two bottlenecks.

Domain Bounding: Domain bounding is implied in the method of capturing the knowledge. A distinct boundary is drawn around each decision that could be made. This boundary encompasses the criteria corresponding to each decision captured in the knowledge base. Thus, the "domain boundary" (as understood in the classical sense) is actually composed of many minute boundaries. Later, you will see how these many small boundaries are kept from conflicting with each other, although some of them overlap.

Knowledge Acquisition: Two kinds of knowledge are captured. The first kind is composed of the decisions themselves. The second is each decision's corresponding criteria. These two kinds of knowledge are entered into a wordprocessor. Overlapping criteria are resolved automatically by the inference engine's preprocessor.

Decisions are based on the current and prior existence of aircraft conditions and other decisions (taken either discretely or statistically accumulated). Each decision is associated with a list of criteria that must be satisfied before its related recommendation or action is delivered. Each criterion has an associated "history" factor. This factor is a period of past time, counting back from the present time, over which the given criterion may have been satisfied. For instance, it may be important to know whether a switch was in the "On" position during the past 5 seconds. The "history" factor can be specified as greater than zero if history is important or made equal to zero if only the present condition of the criterion is needed. A mechanism for capturing and organizing this type of knowledge can take the form of a wordprocessor's input screen and its editing, deleting, and inserting functions.

There are three elements to a decision description: decision name, ON tasks (those tasks to be performed when the decision is made), OFF tasks (those tasks to be performed when the decision is no longer appropriate), and the criteria list (what criteria must be met before the decision is appropriate; these are the system states that must be active).

Decision specifications are in the following format. Note that "~" indicates a not-condition. A not-condition is true if the condition is not active. In the present implementation, ON and OFF tasks are displayed messages. These could ultimately become lists of tasks to be scheduled by a real-time operating system. Figure 12 illustrates the input format for each decision. (In this case, we are assuming that HISTORY = 0.)

```
 decision name (one name on a single line, spaces allowed)
 on tasks      (name list on a single line, separated by spaces)
 off tasks     (name list on a single line, separated by spaces)
 criteria list (one name on a single line, spaces allowed, this
               list may take any number of lines)
 \             (ends the criteria list & decision specification)
... next decision specification follows
end-of-file signals end of knowledge base
```

Figure 12. Input format for capturing recommendations and actions with their associated criteria (Note: Any number of decisions and criteria can be listed.)

To apply the knowledge base input format, we can draw several decisions and related criteria from the aircraft scenerio given in Figure 3.

a) When the generator failure begins, the electrical system, epu generator, main generator, and master caution lights come on.

b) An audible master caution warning occures.

c) The pilot is offered recovery procedures.

d) If recovery does not occure in a certain length of time, the pilot is offered landing procedures.

Figure 13 shows how the appropriate rules and criteria could be captured in the knowledge base. (˜ refers to NOT.) Figure 14 shows a sample result from the use of this knowledge base.

```
Master Caution
master
n_master
master caution light
\
Main generator failure
m_gen
n_mgen
main gen light
\
EPU generator failure
e_gen
n_e_gen
epu gen light
\
Main and EPU generator failure
me_gen
n_me_gen
main gen light still on
epu gen light still on
\
Electrical system failure
electr
n_electr
elec sys caution light
\
EPU/Main Compensation
emcomp
n_emcomp
epu run light
main gen light
~epu gen light
\
Ready for Taxi
taxi
n_taxi
landing gear down
~main gen light
~epu gen light
~epu run light
~master caution light
~acft batt to flcs light
~acft batt depleting light
~arresting hook down
~elec sys caution light
\
```

```
Attempt EPU Switch
epu_sw
n_epu_sw
main gen light
epu gen light
~epu generator switch on
\
Throttle Advance
throttle
n_throtl
main gen light
epu gen light
epu generator switch on
~main gen light still on
~epu gen light still on
single throttle advance
\
Battery Depleting Timer: 5 Min
batt05
n_batt05
main gen light still on
epu gen light still on
batt depleting 5 min
\
Battery Depleting Timer: 10 Min
batt10
n_batt10
main gen light still on
epu gen light still on
batt depleting 10 min
\
Generic Landing (Electrical)
e_land
n_e_land
landing required (elec)
~main gen light still on
~epu gen light still on
\
Cycle Main Power Switch
main
n_main
main gen light
epu gen light
epu generator switch on
~main gen light still on
~epu gen light still on
single throttle advance
airspeed up for 10 sec
\
```

Figure 13. Example knowledge base of specific decisions and their criteria

```
READING AIRCRAFT CONDITION: main gen light AT TIME = 486
FIRING RULE: Main generator failure

**** EXECUTE TASK: m_gen ****

READING AIRCRAFT CONDITION: elec sys caution light AT TIME = 488
FIRING RULE: Electrical system failure

**** EXECUTE TASK: electr ****

READING AIRCRAFT CONDITION: landing required (elec) AT TIME = 488
FIRING RULE: Generic Landing (Electrical)

**** EXECUTE TASK: e_land ****

READING AIRCRAFT CONDITION: master caution light AT TIME = 490
FIRING RULE: Master Caution

**** EXECUTE TASK: master ****

READING AIRCRAFT CONDITION: epu gen light AT TIME = 496
FIRING RULE: EPU generator failure

**** EXECUTE TASK: e_gen ****

FIRING RULE: Attempt EPU Switch

**** EXECUTE TASK: epu_sw ****

READING AIRCRAFT CONDITION: epu generator switch on AT TIME = 501
DEACTIVATING RULE'S TASK EXECUTION LIST: Attempt EPU Switch

**** EXECUTE TASK: n_epu_sw ****

REMOVING TASK epu_sw FROM TASK EXECUTION LIST
READING AIRCRAFT CONDITION: single throttle advance AT TIME = 513
FIRING RULE: Throttle Advance

**** EXECUTE TASK: throttle ****

READING AIRCRAFT CONDITION: airspeed up for 10 sec AT TIME = 521
FIRING RULE: Cycle Main Power Switch

**** EXECUTE TASK: main ****

READING AIRCRAFT CONDITION: airspeed up for 10 sec\ AT TIME = 565
DEACTIVATING RULE'S TASK EXECUTION LIST: Cycle Main Power Switch

**** EXECUTE TASK: n_main ****
```

```
REMOVING TASK main FROM TASK EXECUTION LIST
READING AIRCRAFT CONDITION: main gen light still on AT TIME = 665
DEACTIVATING RULE'S TASK EXECUTION LIST: Throttle Advance

**** EXECUTE TASK: n_throtl ****

REMOVING TASK throttle FROM TASK EXECUTION LIST
DEACTIVATING RULE'S TASK EXECUTION LIST:
  Generic Landing (Electrical)

**** EXECUTE TASK: n_e_land ****

REMOVING TASK e_land FROM TASK EXECUTION LIST
READING AIRCRAFT CONDITION: acft batt depleting light AT TIME = 674
READING AIRCRAFT CONDITION: epu gen light still on AT TIME = 675
FIRING RULE: Main and EPU generator failure

**** EXECUTE TASK: me_gen ****

READING AIRCRAFT CONDITION: batt depleting 5 min AT TIME = 974
FIRING RULE: Battery Depleting Timer: 5 Min

**** EXECUTE TASK: batt05 ****

READING AIRCRAFT CONDITION: arresting hook down AT TIME = 974
READING AIRCRAFT CONDITION: batt depleting 5 min\ AT TIME = 1280
DEACTIVATING RULE'S TASK EXECUTION LIST: Battery Depleting Timer:
  5 Min

**** EXECUTE TASK: n_batt05 ****

REMOVING TASK batt05 FROM TASK EXECUTION LIST
READING AIRCRAFT CONDITION: batt depleting 10 min AT TIME = 1280
FIRING RULE: Battery Depleting Timer: 10 Min

**** EXECUTE TASK: batt10 ****

....... scenario continues until landing is completed
```

Figure 14. Example result from exercising the criteria/decision knowledge base.

It is well worth your while to perform careful quality control on the knowledge base prior to its use. Basic checks such as spelling, use of criteria that are not defined conditions or decisions, decisions without criteria, and duplicate descriptions of decisions should be made at a minimum. Once the information on decisions and their criteria has been captured, the resulting knowledge base needs to be restructured for use by the inference engine.

The inference engine's interaction with the pilot could also include decision rational. A glossary can contain any amount of information related to the decisions. The nodes in the decision matrix could then point to entries in this glossary for retrieval at the appropriate moment. Such a glossary also could contain help information, such as detailed checklists and other displays, to support pilot queries. Such a capability has not been implemented at this time since conversations with pilots have indicated that such a capability would not be useful.

5.0 FORMULATING THE DECISION/CRITERIA EXPERT SYSTEM

This section discusses a data structure and algorithm to implement decision/criteria expert systems. The inferencing method developed is designed to enable system expandability via parallel processing to ensure real-time performance.

The formulation for the evaluation of decision/criteria knowledge bases depend upon a criteria vector, a response (action) vector, and a set of relationships between criteria and responses. The relationships between decision criteria and responses are traditionally expressed as rules in the form:

```
IF   (all listed criteria are true)
THEN (perform all appropriate actions)
```

Although we will use the term "rule," it is not necessary for domain experts to be burdened with the syntax and terminology of traditional expert systems rules. As shown in Section 4, the "rules" can be simply collected as lists. This present section shows what goes on in the computer to make use of those lists.

The criteria vector **c** is a vector of m Boolean (True or False) variables. These criteria are the aircraft conditions discovered by the condition analysis expert system discussed earlier. For example, criterion c_{33} might represent the current amount of fuel (raw data source s_{12}) vs. a minimum fuel reserve. For instance, $c_{33} = [s_{12} \leq f_R]$) and c_{33} is True when there is insufficient fuel reserve.

A set of n rules define a rule vector **r**, relating the criteria and response vectors, defines the on-board expert system that will advise the pilot and, with the pilot's consent, act on his/her behalf. Each rule can be formulated in terms of a conjunction of simple Boolean criteria that lead to a set of actions. If all a given rule's criteria are true, a given action will result. (Note that an "action" could be composed of any number of activities.) This action could either be something that is automatically performed for the pilot or it could be a recommendation to the pilot. All possible actions define an action vector **a** of size p. Each rule is expected to involve only a relatively small number of m possible criteria. For example, each rule may have up to 10 criteria. The rule-base is built off-line, and not modified during the evaluation process. For example, a typical rule might look like this (remember that "˜" means "NOT"):

Rule r_{123}: $a_{12} \Leftarrow\!= c_1$ & $\tilde{}c_5$ & c_6 & c_{18} & $\tilde{}c_{47}$ & $\tilde{}c_{99}$

This rule is interpreted as stating that action a_{12} will be taken if criteria c_1, c_6, and c_{18} are all true while c_5, c_{47}, and c_{99} are all false.

In a typical rule-based expert system, the inference engine performs three standard operations:

1) the match operation matches the criteria against the rules to see which actions could occur

2) the resolve operation chooses which of these actions will actually occur

3) the execute operation actually generates the appropriate actions and updates working memory

It is assumed that no action in vector **a** directly alters the criteria vector **c** in any way at any time-step t_k. Further, it is possible that different rules can include the same action. Hence, by expressing each rule only in terms of simple AND and NOT logic, it's evaluation can be done very efficiently and independently. (Note that OR constructs are equivalent to multiple rules that specify the same action.)

Duplicate actions are prevented by the action triggering mechanism that is external to the inference engine described in this paper. This mechanism sets a "triggered" flag when the action is started. In a given update cycle, this flag can only be set once. All other attempts to set this flag are ignored. When the action is completed, the flag is reset.

Conflicting actions also can be resolved by expanded criteria such as "$\tilde{}a_{51}$." This means that the rule's consequent action would not take place if action #51 is underway. This technique would mean that the associated data structure would have to be updated for each action start and completion. Thus, the action vector would have an associated "action triggered" vector. This could be accomplished by simply making the element in the action vector negative for "action underway" or positive for "action not underway."

5.1 DECISION/CRITERIA DATA STRUCTURE AND ALGORITHM DEVELOPMENT

The data structure and algorithm developed to implement this decision/criteria expert system are designed for a single fast processor or parallel processors with a correspondingly slower clock-speed. The data structure uses the notion of a blackboard that contains the raw data source and criteria vectors described above. In addition, other vectors serve as index vectors to completely define the knowledge base.

A blackboard is a global and dynamic database for the communication of independent asynchronous information for related aspects of a given problem. The aircraft system blackboard will contain the criteria vector **c**. The truth values of this vector will be updated by the previously described condition analysis inference engine at each time-step. Each update of the vector **c** will immediately initiate a new evaluation of the rules' criteria. So, the evaluation of the entire knowledge base must be completed during time-step t_k before the criteria vector is updated at time-step t_{k+1}.

Two algorithms for the evaluation of the decision/criteria knowledge base will now be presented. Method-1's criteria-oriented approach takes the view that a large number of criteria change during each time-step. Method-2's rule-oriented approach takes the opposite view, assuming that only a few criteria will change each time-step. It is helpful to have an understanding of both methods.

5.1.1 METHOD-1

The simplest method for an expert system evaluation assumes that the rules and their criteria are listed in priority order. This is equivalent to a priority-oriented backward chaining method. This is the obvious choice when n << m and no other assumptions are made about available data. (Note that if these rules were not prioritized, then this first algorithm could be viewed as a forward chaining algorithm.) Because no OR-logic is present in a given rule, Method-1 stops with the first c_i = False (or first c_i = True in the case of $\tilde{}c_i$). If these rules were ranked and evaluated from highest to lowest priority, then the first action produced (if any) would be the most important from the pilot's point of view. If required, different levels of parallelism could be employed during this evaluation process. If the processing time is not fast enough, then rules having the same priority could be grouped according to their number of criteria to equalize the work among the parallel processors, as discussed by Tout and Evans. A simple example of a rule-base with four rules is:

Rule r_1: $action_1$ <== c_1 & c_3 & $\tilde{}c_4$ & c_{40} & $\tilde{}c_{98}$ & c_{99}
Rule r_2: $action_1$ <== c_2 & c_4 & c_{22} & $\tilde{}c_{85}$
Rule r_3: $action_2$ <== c_5 & c_{99}
Rule r_4: $action_3$ <== c_1 & c_{50}

Note that r_1 is the highest priority rule and r_4 is the lowest priority rule. The criteria are evaluated left to right. Evaluation stops as soon as a False is detected. The left-to-right evaluation can be thought of as assuming that the left-most criteria are expected to occur most often and are thus evaluated first.

These rules could be represented efficiently by using three vectors: the previously discussed action vector **a** whose elements each point to a specific task to be completed, a <u>query vector</u> **q**, identifying which criteria have to be checked, and an <u>index vector</u> **End**, that delimits the criteria that appear in each rule. For the above rule-base, consider:

Rule 1: $\text{action}_1 = a_1$; $q_1 = 1$, $q_2 = 3$, $q_3 = -4$, $q_4 = 40$, $q_5 = -98$, $q_6 = 99$; so
$\text{Start}_1 = 1$; $\text{End}_1 = 6$
Rule 2: $\text{action}_2 = a_1$; $q_7 = 2$, $q_8 = 4$, $q_9 = 22$, $q_{10} = -85$; so
$\text{Start}_2 = 7$; $\text{End}_2 = 10$
Rule 3: $\text{action}_3 = a_2$; $q_{11} = 5$, $q_{12} = 99$; so
$\text{Start}_3 = 11$; $\text{End}_3 = 12$
Rule 4: $\text{action}_4 = a_3$; $q_{13} = 1$, $q_{14} = 50$; so
$\text{Start}_4 = 13$; $\text{End}_4 = 14$

Here **q** employs positive integers to indicate criteria indices. Negative indices indicate criteria complements (NOT-criteria). Note also that Rules 1 and 2 have the same consequent. From this example, one has the 14-element query vector:

q:	1	3	-4	40	-98	99	2	4	22	-85	5	99	1	50

This allows for direct and very fast access to the **c** vector stored on the blackboard (only one internal integer multiplication and addition are needed to compute any cell address). If parallel processors are used, this Boolean criteria vector **c** can be accessed from the blackboard by all processors. If multicomputers are used,

c would be communicated to the local memory of each processor and this communication time will need to be considered, according to Lester. Each processor also must use components from the query vector **q**. Note the relationship $\text{Start}_{j+1} = \text{End}_j + 1$ with $\text{Start}_1 = 1$, so only the **End** unsigned integer index vector is needed by the algorithm. In this example, one has:

End: | 6 | 10 | 12 | 14 | which implies **Start:** | 1 | 7 | 11 | 13 |

Note also that vector **q** has a number of elements equal to the sum of the number of criteria queried by each rule. An upper-bound for this number is **m**. Vector **End** has n elements, the total number of rules.

This method yields Algorithm-1, presented below, which is a relatively simple and straightforward algorithm that can use these data structures.

```
Forall i := 1 to n do in parallel
   begin
      if i = 1
         then j := 1
         else j := $End_{i-1}$ + 1;
      Fired := TRUE;
      while j $\leq$ $End_i$ and Fired do
         begin
            k := $q_j$;
            if k $\geq$ 0 and not $c_k$ then
               Fired := FALSE
            else if k < 0 and $c_{-k}$ then
               Fired := FALSE;
            j := j + 1
         end;
      if Fired then perform action $a_j$
   end
```

In Algorithm-1, the Forall statement creates up to n parallel processes. If p is the number of parallel processors and $p \geq n$, then this loop completes as soon as the slowest of these processes finishes execution. Since each processor takes time to evaluate only one rule, the total parallel processing time at a given time-step is the maximum single-rule evaluation time. If $p < n$, then the next available processor would evaluate the next unprocessed rule. Hence the total parallel processing time at a given time-step is then the maximum of all sums of the individual processor rule evaluation times. Notice that this reduces to a normal sequential processing algorithm when $p = 1$.

For example, if $p \geq 4$ and it takes an estimated average of 50 microseconds to check each criterion in the previous 4-rule situation, then 4 copies of the loop body will be created on 4 different processors, each with its own value of the loop control i-variable. These will execute in parallel with respective times of 300, 200, 100, 100 microseconds, at most (as soon as a FALSE is determined, the process stops for the current rule). This would then take at most 300 microseconds in parallel versus at most 700 microseconds if done sequentially, giving a speedup of 7/3 or approximately 2.3. Here the action performance time (e.g., displaying an information screen) was not considered, nor was processor-assignment overhead or communication time. Of course, any of these three times can have a significant effect on this expert system evaluation process. Typically, the actions resulting from criteria satisfaction are performed by a separate processor suite.

5.1.2 METHOD-2

The previous method does not have the advantage of searching in any informed way whenever the raw data (and hence a criterion) changes, because the indexing is in the opposite direction from rule to criterion. A second, combined forward-backward chaining method, could be used to check only the rules whose criteria values have changed since the last evaluation of the rule-base. To do this, one also could index in the opposite direction, checking only the rules having newly changed (currently "active") criteria. The forward phase identifies the changed criteria and rules that use these criteria. The backward phase is the same as before with presumably fewer rules to process. For example, using the same four rules as before, one could have something like:

Criterion c_1: NeedToCheck_1 = False; $\text{First}_1 = 1$; $\text{Last}_1 = 2$; $r_1 = 1$, $r_2 = 4$
Criterion c_2: NeedToCheck_2 = True; $\text{First}_2 = 3$; $\text{Last}_2 = 3$; $r_3 = 2$
Criterion c_3: NeedToCheck_3 = False; $\text{First}_3 = 4$; $\text{Last}_3 = 4$; $r_4 = 1$
Criterion c_4: NeedToCheck_4 = True; $\text{First}_4 = 5$; $\text{Last}_4 = 6$; $r_5 = 1$, $r_6 = 2$
Criterion c_5: NeedToCheck_5 = False; $\text{First}_5 = 7$; $\text{Last}_5 = 7$; $r_7 = 4$
Criterion c_6: NeedToCheck_6 = True; $\text{First}_6 = 0$; $\text{Last}_6 = 0$;
not in any rule

....

Criterion c_{99}: NeedToCheck_{99} = False; $\text{First}_{99} = 13$; $\text{Last}_{99} = 14$; $r_{13} = 1$, $r_{14} = 3$

Assuming criteria c_2, c_4, and c_6 were the only ones whose truth value changed, their **NeedToCheck** components would be set to True in the blackboard. This would cause $Rule_2$, $Rule_1$, and $Rule_2$ to be consolidated into the set { $Rule_1$, $Rule_2$ }. Components $NeedToCheck_2$, $NeedToCheck_4$, and $NeedToCheck_6$ are reset to False after those rules had been re-evaluated. The **First** and **Last** vectors are similar to the **Start** and **End** vectors of Method-1. Their indices point to blocks of rules listed in vector **v**. The idea is for criteria to point to their rules. Then it is possible to re-evaluate only those rules for which criteria truth values have changed.

The efficiency of this method is related to the number of criteria whose truth value change at any time-step. The number of criteria that change at any time-step is highly dependent upon the application. The fewer the criteria that change, the faster this method will be.

Each change in the raw data source vector **s** at time-step t_k can cause the truth value of the Boolean criteria vector **c** (and its corresponding **NeedToCheck** vector) to change. Each criteria vector change, in turn, causes a set (or prioritized list) of rule numbers to be defined. Each rule in the set would contain at least one changed criteria and only the rules in this set need to be checked to see if all criteria hold. Once these rules are identified, the actual criteria checking occures as in Algorithm-1. It is possible to go further and only check the previously unsatisfied criteria in those rules. However, the additional software complexity, memory utilization, and execution time would likely exceed any savings compared to simply using Algorithm-1.

Figure 15 summarizes the previous discussion of both methods. In order to map the rules to actions, each element in the Rules -> Action vector contains a pointer to an element in the Action vector. A record-oriented data structure also can be used to implement this system.

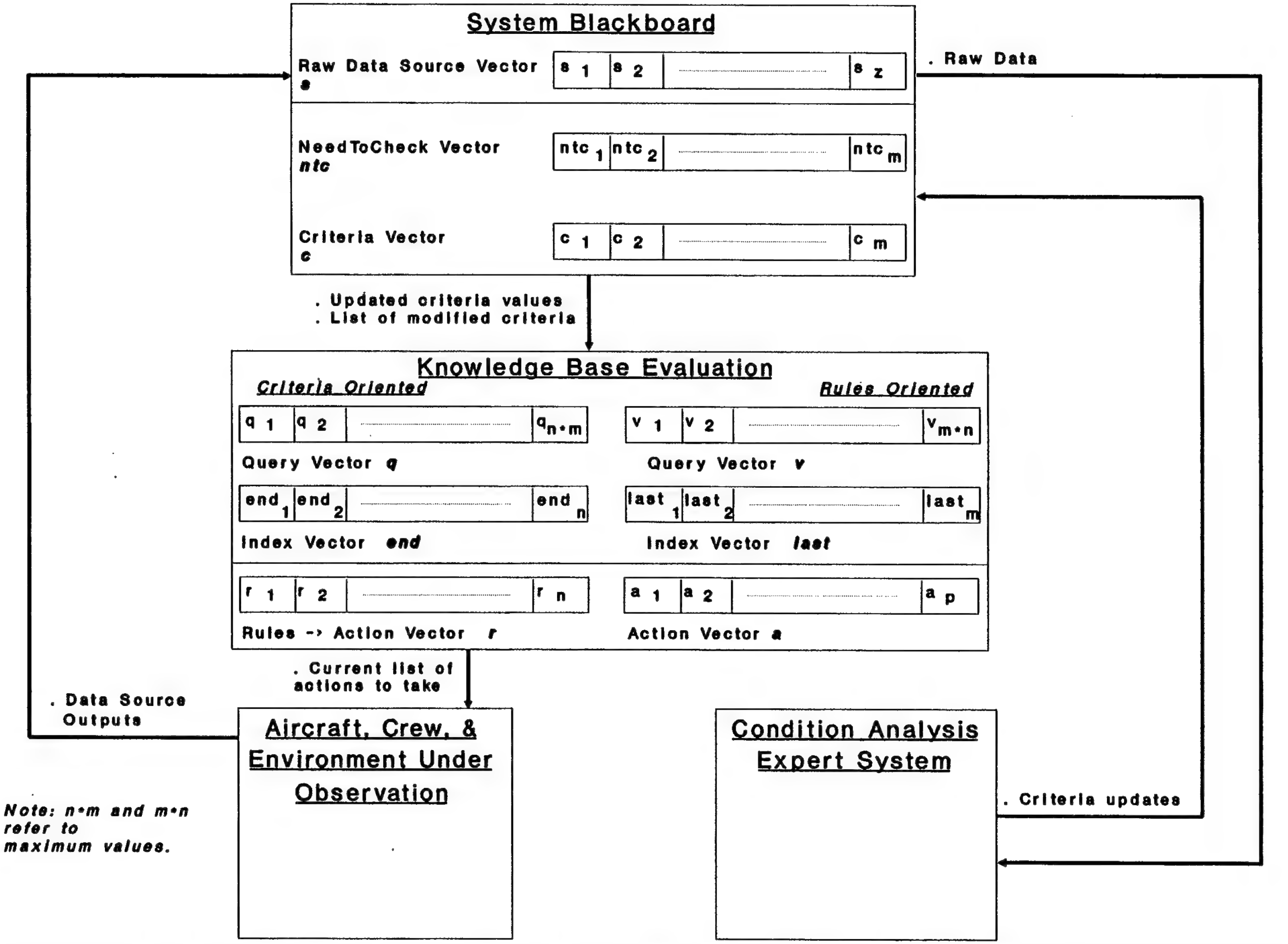


Figure 15. Decision/Criteria Expert System Operation Summary

5.2 DEALING WITH UNCERTAINTY

In practice, one or more sensor failures may lead to undetermined (uncertain) components of the raw data source vector **s**, which may lead to one or more unknown truth values in the criteria vector **c**. For every rule, one of three situations must hold at time-step t_k:

1) the truth value of all its criteria are known

2) there are criteria with unknown truth values, but at least one of the known criteria fails to be satisfied

3) all of the known criteria are satisfied, but there are still criteria of unknown truth value

The first two situations are easily addressed, since it can be exactly determined whether or not the rule will fire. In the first case, the rule will fire. In the second case it will not fire. In the third situation, criteria with unknown truth values determine whether the rule will fire or not. Because of the possible interdependence of criteria, it is very difficult to determine any type of formal probability or level of certainty measure associated with the firing of this rule since multivariable conditional probabilities are involved. However, it is possible to report a possible action by simply keeping count of the number of criteria that are unknown for the given rule. This requires that each component of the criteria vector **c** have one of three truth values (True, False, Unknown), instead of just True or False as used in Algorithm-1 and Algorithm-2. A possible action occurs if a rule's criteria are either True or Unknown.

The algorithm to do this is a variation of Algorithm-1, but is slightly more complex and takes more processing time. This is because an additional IF-test is needed, and two additional counting operations are necessary for reporting when one or more of the necessary **c** truth values are unknown. The reporting of the Ucount/Ncrit ratio is intended to give the pilot some measure of exactly how many unknown criteria (Ucount) exist relative to the total number of criteria (Ncrit) that are used in the given rule. For example, if there are 10 criteria in the rule and a possible action is reported with a ratio of 1/10, then the pilot might place more confidence in it than if a ratio of 7/10 was presented. The smaller the ratio, the more confidence is justified. Of course, one could reverse this reasoning by using (1.0 - (Ucount/Ncrit)).

The algorithm designed to deal with this uncertainty is presented below as Algorithm-1u:

```
Forall i := 1 to n do in parallel
   begin
      if i = 1
         then j := 1
         else j := $End_{i-1}$ + 1;
      Fired := TRUE;
      Ncrit := 0;
      Ucount := 0;
      while j $\leq$ $End_i$ and Fired do
         begin
            k := $q_j$;
            if $c_{|k|}$ is Unknown then
               Ucount := Ucount + 1
            else if k $\geq$ 0 and $c_k$ is False then
               Fired := FALSE
            else if k < 0 and $c_{-k}$ is True then
               Fired := FALSE;
            j := j + 1;
            Ncrit := Ncrit + 1
         end;
      if Fired then
         if Ucount = 0
            then perform action $a_j$
            else report possible $a_j$ with Ucount/Ncrit ratio
   end
```

This algorithm also could be modified to report exactly which unknown criteria caused the problem. When considered in the total application context, it also may be useful to report the failed sensors that caused the unknown criteria.

6.0 Physical Implementation

The two expert systems for pilot/vehicle interfaces discussed in this paper fall into the general class of decision support systems. Sprague and Carlson provide an excellent discussion of that subject. Decision support systems are usually discussed in the light of business applications. However, the same principles apply to the embedded requirements of aircraft.

Figure 16 illustrates the overall architecture envisioned for integration of the two expert systems into an aircraft's embedded automation. The expert systems comprise the decision support block. Data fusion is performed by the condition analysis expert system. Condition fusion and task generation are performed by the decision/criteria expert system. "Fusion" refers to the act of bringing together independent items so that a bigger picture of the whole may be created. The value achieved is that raw data and operational knowledge are used to determine tasks to be performed by the automation.

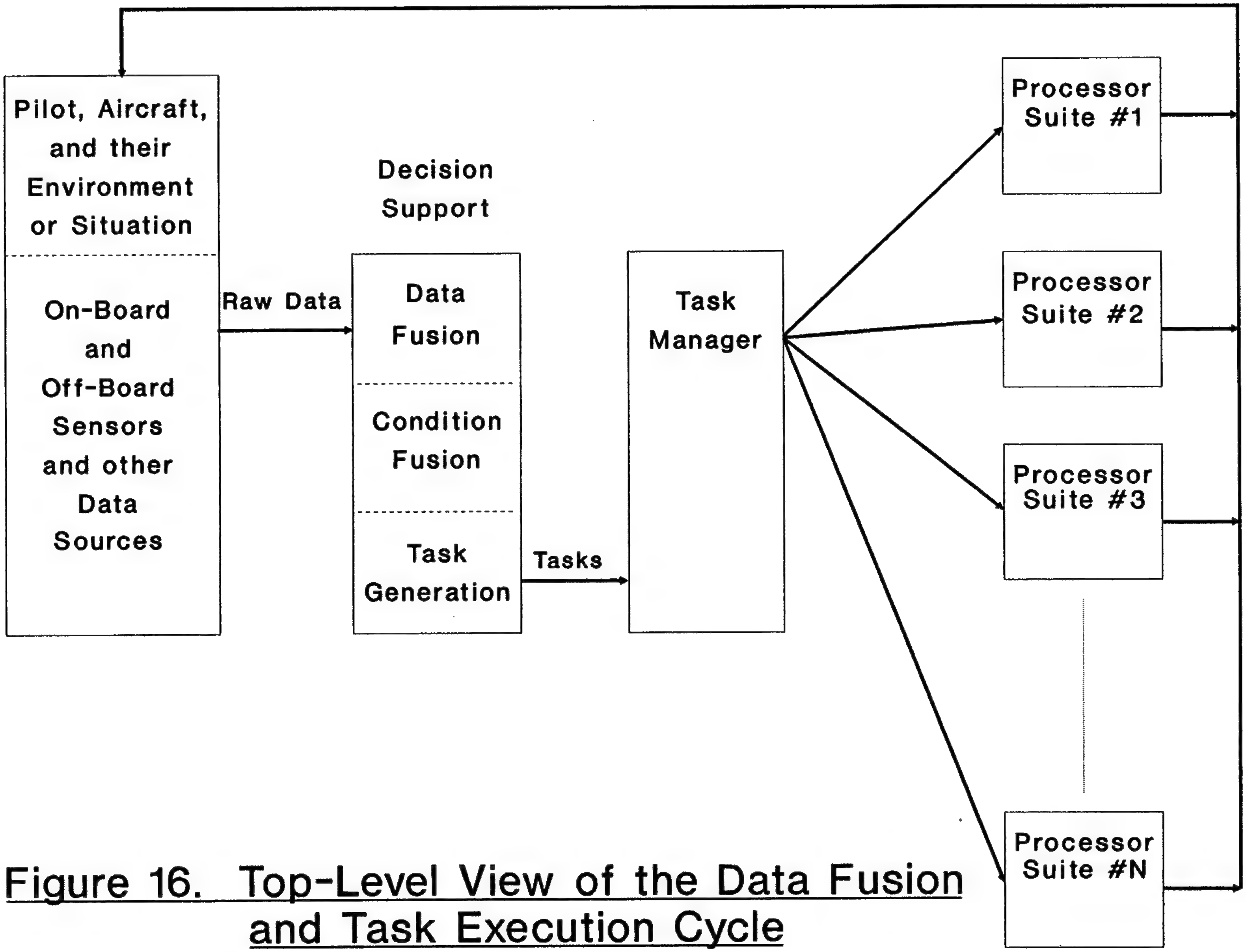


Figure 16. Top-Level View of the Data Fusion and Task Execution Cycle

Figure 17 carries this thinking one step further and represents the software generated in support of this project. The scenario generator is used in place of physical equipment. It allows the two expert systems to be tested with raw data similar to that obtained directly from hardware. At this point in the project, no feedback loop exists. That will come later when the expert systems

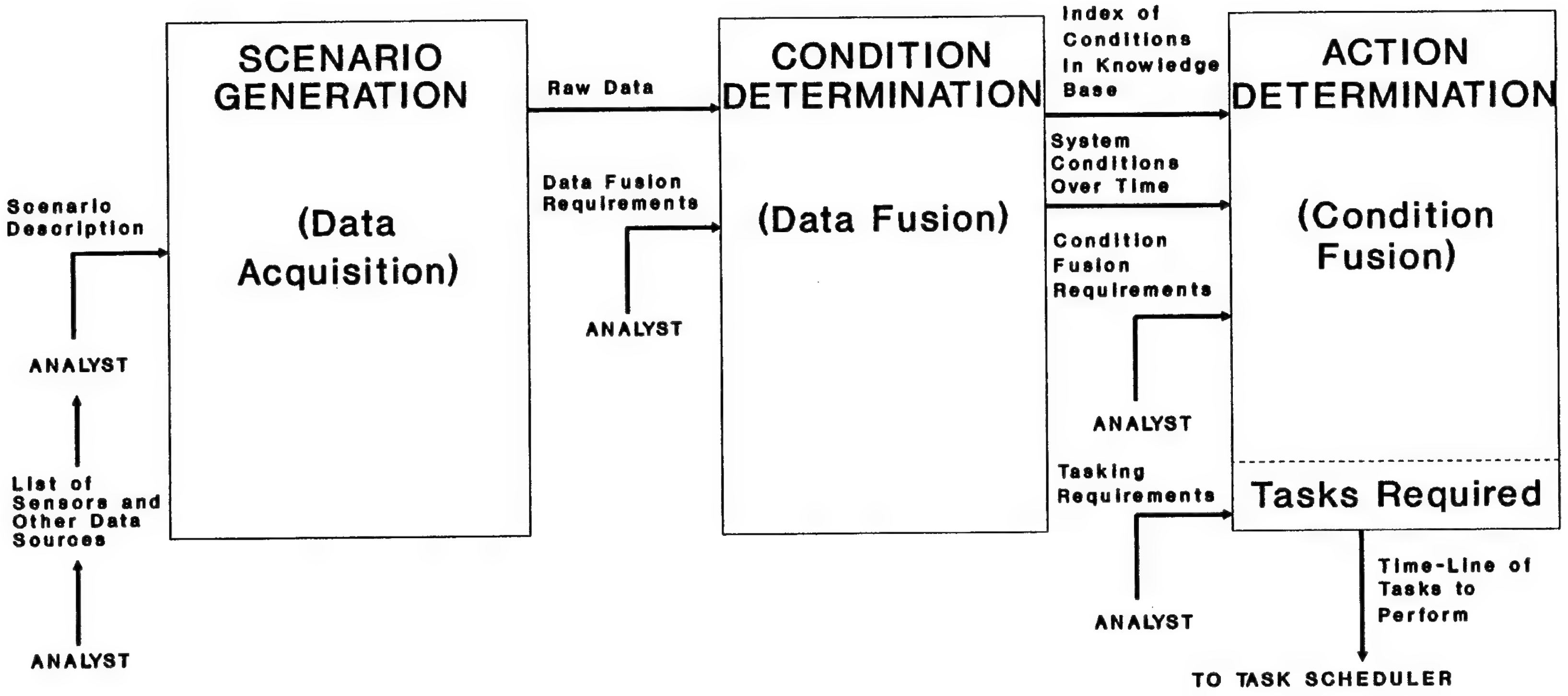


Figure 17. Overview of Software for Decision Support

are integrated into a cockpit evaluation system. The feedback loop is created when the cockpit evaluation system generates raw data based on its operation in light of the tasks running in the system.

There are two ways that parallel operation is achieved by this implementation. First, there are several processor suites that can accommodate the tasks being generated. Second, each processor suite is composed of several processing units. Parallel architectures permit not only several disparate tasks to be processed at the same time but also accelerate individual tasks though the use of more than one processor to perform the task. The inherent parallel nature of both expert systems fits well into this style of design.

The software for the expert systems has been written in both Pascal and Ada. Current testing involves simulated raw data, knowledge bases derived from operational personnel and aircraft technical manuals, and is performed on a DOS PC. After the current tests, integration with a cockpit evaluation system will take place. This system is based on Silicon Graphics computers under Unix.

7.0 CLOSING COMMENTS

Implementing the expert systems technology described in this report requires much careful thought and attention to detail. For instance, you must decide what the expert system is to do when no aircraft conditions are active. Perhaps you will want an "all normal" condition for each parameter or at least a "no active condition" with an appropriate recommendation. As you can see from this paper, developing an expert system to support a pilot/vehicle interface takes a lot of time and an in-depth understanding of the aircraft. It goes without saying that the expert system will only be as smart or comprehensive as the domain expert makes it.

Something else to be aware of is that the techniques described here will not detect novel situations, situations the expert system has not been specifically trained for. For that kind of generalization, neural network and fuzzy logic technology may be appropriate. These technologies could at least be used to preprocess the raw data before it reaches the conditional analysis inference engine. The expert system methods described here can only generalize in the sense that value bounds instead of specific values can be entered into the knowledge base. These methods are appropriate for automating the types of information found in aircraft technical orders and aircrew checklists.

Fuzzy logic is an extension of the classical logic employed by this paper. There is nothing "fuzzy" about fuzzy logic. It is, rather, founded on an exacting mathematics and may be thought of as "continuous state" logic, as opposed to "discrete state" logic. While neural network technology can be used with expert systems, it is not at all based on the same principles, being an example-based technology rather than a knowledge-based technology. For more on neural networks, see the works of Jorgensen & Matheus, Lippmann, Rumelhart & McClelland, and Reece. Both Lippmann and Rumelhart & McClelland provide numerous examples. Fuzzy sets are discussed by Dubois & Prade, Kaufmann & Gupta, Zimmerman, and Novak. Dubois & Prade and Zimmerman both provide extensive examples.

Be cautious about letting a system like this (or any automatic system) run on its own without some human supervision. It is difficult to build deep knowledge such as common sense into an automated system. Also, the knowledge base probably will not be as comprehensive as the experience of a human expert. We cannot assume that, by definition, an expert system has more knowledge or experience than a human. Building and updating an expert system is as important and time consuming as training and updating a human. The difference is in consistency of knowledge use and retention, with expert systems excelling in retention and consistency. On that note, you also must take care to not let conflicting information creep into the knowledge base, a common occurrence when more than one domain expert participates. For some good reading on this subject see the reports by Rolandi and Marcot. Tennenbaum and

Augenstein offer a good fundamental book on developing and using data structures, the heart of expert systems technology.

One might comment that the techniques described here sound a lot like AND/OR logic that could be implemented in hardware or via traditional sequential code. This is true for specific and unchanging condition and decision cases. Bear in mind, however, that the complexity of construction in hardware and sequential code increases with the complexity of the conditions' specifications, especially conditions that are time dependent and based on combinations of samples from several sensors. With expert systems techniques, one need only take the trouble to fill in a spreadsheet screen implemented on a wordprocessor. This permits going from concept to reality very fast. Further, additional problems come up when one tries to modify hardware and sequential code to add conditions or decisions, to change existing ones, or to remove existing ones. Sequential code needs a good programmer. Hardware needs a hardware expert. Expert systems need only a domain expert who can do simple data entry to a wordprocessor screen; data that is natural to the domain expert and in units he/she normally deals with.

8.0 SUMMARY

In this report we started with the idea that a large mass of real-time data is needed to manage modern aircraft and that humans need help coping with task and data overload. We have described expert systems techniques for using raw data to indicate aircraft conditions that lead ultimately to recommendations and actions. Methods for automatically generating information networks and decision matrices from knowledge bases were shown. We have seen how important it is to have a knowledge acquisition method that is unconstrained by syntax and special terminology. An orientation to the domain expert is essential. The example we followed was the aircraft pilot/vehicle interface but these techniques are useful for working with any time sampled system. Finally, we have looked at specific steps for putting these techniques into practice. The authors have developed code in Pascal and Ada to implement the ideas discussed in this paper. Their next step is integration with a cockpit evaluation system for pilot-in-the-loop part-task testing.

9.0 BIBLIOGRAPHY

Dubois, Didier and Henri Prade Fuzzy Sets and Systems: Theory and Applications, New York, NY: Academic Press, 1980

Emerson, Terry and John Reising "The Effect of an Artificially Intelligent Computer on the Cockpit Design Paradigm", Proc: 2nd Joint GAF/RAF/USAF Workshop on Human-Electronic Crew Teamwork, Alexandria, VA: National Technical Information Service, Report # WL-TR-92-3078, Jul 92

Gal, Annie and Jack Minker "A Natural Language Database Interface that Provides Cooperative Answers" IEEE Conference on Artificial Intelligence Applications, 1985

Harvey, David S. "V-22 Crash Prompts Questions About Cockpit Data and Displays", Avionics, Aug 93

Jorgensen, Chuck and Chris Matheus "Catching Knowledge in Neural Nets", AI Expert, Dec 86

Kaufmann, Arnold and Madan M. Gupta Introduction to Fuzzy Arithmetic, New York, NY: Van Nostrand Reinhold, 1985

Lester, Bruce P., The Art of Parallel Programming, Prentice Hall, Englewood Cliffs, NJ, 1993

Lippmann, Richard P. "An Introduction to Computing With Neural Nets", IEEE ASSP Magazine, Apr 87

Marcot, Bruce "Testing Your Knowledge Base", AI Expert, Jul 87

Moray, Neville and Johy Lee "Trust and Allocation of Function Between Human and Machine", Proc: ASME Winter Meeting, Dec 91

Novak, Vilem Fuzzy Sets and Their Applications, Philadelphia, PA: Adam Hilger, 1989

Noyes, James L. Artificial Intelligence With Common Lisp: Fundamentals of Symbolic and Numeric Processing Lexington, MA: DC Heath, 1991

Noyes, James L. Expert System Rule-Base Evaluation Using Real-Time Parallel Processing Alexandria, VA: National Technical Information Service, Report # WL-TR-93-3098, 1993

Pawlowski, Thomas J. and Christine M. Mitchell "Direct Manipulation to Facilitate Supervisory Control and Intent Inferencing in Complex Dynamic Systems", Proc: IEEE Int Conf on Systems, Man, and Cybernetics, Oct 91

Raeth, Peter, G.; James L. Noyes; and Anthony J. Montecalvo "A Scaleable Framework for Adding Crisp Knowledge to Pilot/Vehicle Interfaces" Invited Paper, IEEE International Conference on Systems, Man, and Cybernetics, Oct 94

Raeth, Peter, G.; James L. Noyes; and Anthony J. Montecalvo "Cockpit Decision Aids Via Trust-Enhancing Knowledge-Based Techniques" Invited Paper, Joint GAF/RAF/USAF Workshop on Human-Electronic Crew Teamwork, Cambridge, England, Sep 94

Raeth, Peter G. (editor and contributing author) Expert Systems: A Software Methodology for Modern Applications Washington, DC; IEEE Computer Society Press; 1990

Raeth, Peter G. "Expert Systems in Process Observation and Control", AI Expert, Sep/Dec 90

Raeth, Peter G. "An Expert Systems Approach to Decision Support in a Time-Dependent, Data Sampling Environment (A Brief Discussion)", IEEE National Aerospace and Electronics Conference, 1988

Reece, Peter "Perceptrons and Neural Nets", AI Expert, Jan 87

Rolandi, Walter G. "Knowledge Engineering in Practice", AI Expert, Dec 86

Rumelhart, David E. and James L. McClelland Parallel Distributed Processing (Vol I, II, & III), Cambridge, MA: MIT Press, 1986

Scott, William B. "F-15E's Night Attack Capability Assessed in Low-Level Flight", Aviation Week and Space Technology, Apr 90

Sheridan, T.B. "Supervisory Control" in G. Salvendy (ed) Handbook of Human Factors, New York, NY: Wiley, 1988

Sprague, Ralph H. and Eric D. Carlson Building Effective Decision Support Systems, Englewood Cliffs, NJ: Prentice-Hall, 1982

Tenenbaum, Arron M. and Moshe J. Augenstein Data Structures Using Pascal; Englewood Cliffs, NJ; Prentice Hall; 1981

Tout, K. R. and D. J. Evans, "Parallel Forward Chaining Technique with Dynamic Scheduling, for Rule-Based Expert Systems," Parallel Computing, Vol 18, #8, pp 913-930, Aug 1992

Trippi, Robert R. and Efraim Turban, "Parallel Processing and OR/MS", Computers & Operations Research, Vol 18, #2, pp 199-210, 1991

Wickens, C.D. Engineering Psychology and Human Performance, Columbus, OH: Merill, 1984

Zimmerman, Hans J. Fuzzy Set Theory and Its Applications (Vol I & II), Hingham, MA: Kluwer-Nijhoff, 1985

10.0 AUTHOR INFORMATION

Peter G. Raeth received the B.S. degree in Electrical Engineering from the University of South Carolina in 1979 and the M.S. in Computer Engineering from the Air Force Institute of Technology in 1980. He is a Major on active duty with the US Air Force. He is stationed with Wright Laboratory in Dayton, Ohio and leads the Pilot/Vehicle Interface Technology Development Section. His ten-member organization conducts, manages, and transitions exploratory development in advanced information processing, decision aids, information controls, and display formats for aerial vehicles. He has given lectures and published articles, papers, and reports on the subjects of expert systems and neural networks since 1977. He is editor and a contributing author of the book, "Expert Systems: A Software Methodology for Modern Applications", published by the IEEE Computer Society in 1990. In 1991 he was awarded a research fellowship in neural networks at the University of Dayton Research Institute. He is a volunteer member of the critical review team for Academic Press. His previous assignment was with Headquarters Air Force Materiel Command as Deputy Air Force Industry R&D Manager. Major Raeth is a member of the electrical engineering honor societies Tau Beta Pi and Eta Kappa Nu, the community service honor society Omicron Delta Kappa, IEEE Computer Society, and the Dayton Special Interest Group on Artificial Intelligence. He has twice won the Armed Forces Communications and Electronics Association's gold medal. In 1995 and 1996 he was selected as one of the top 20 technology leaders in the Dayton Ohio region by the Dayton Affiliate Societies Council. Maj Raeth can be reached at the following address:

Wright Laboratory, WL/FIGP
Pilot/Vehicle Interface Branch
2435 Flyway Court
Beavercreek Ohio 45431-4115 USA
ac095@dayton.wright.edu, 513-320-0793

Anthony J. Montecalvo received a BS in Aeronautical and Astronautical Engineering from Ohio State University in 1984. He spent 10 years flying the F-16 Fighting Falcon on USAF active duty and in the Air Force Reserve. He is currently flying the C-141 Starlifter. He is a senior pilot with over 1700 flight hours. Mr Montecalvo is employed by Veda Inc, supporting the Wright Laboratory Pilot/Vehicle Interface Branch. He studies advanced technologies that have the potential to reduce pilot workload and improve situation awareness, and develops pilot/vehicle interface concepts to incorporate these technologies into future combat aircraft. Mr Montecalvo can be reached at the following address:

Veda Incorporated
5200 Springfield Pike Suite 200
Dayton Ohio 45431-1255 USA
513-253-4770

James L. Noyes is a Professor of Computer Science at Wittenberg University. Prior to joining the faculty at Wittenberg, he was a Mathematician, Operations Research Analyst, and civilian Branch Chief for the Air Force at Wright-Patterson AFB. His research interests involve the integration of symbolic and numeric processing and have focused upon neural networks, expert systems, and mathematical optimization techniques. He has been a Senior Research Fellow for both the National Research Council and the Air Force Office of Scientific Research. He recently authored the college textbook: Artificial Intelligence with Common Lisp (published in 1992 by D.C. Heath). Dr Noyes can be reached at the following address:

Wittenberg University
Department of Mathematics and Computer Science
Springfield Ohio 45501-0720
noyes@wittenberg.edu, 513-327-7858

APPENDIX A

A DETAILED EXAMPLE APPLICATION FROM FLIGHT SAFETY

The techniques discussed in this paper were also applied to an expanded version of the flight safety example proposed by Jeannette Lawrence in Introduction to Neural Networks and Expert Systems, Nevada City CA: California Scientific Software, 1992. She observed various aircraft data sources to determine safety of flight. An overview of the data sources is given in Figure A-1.

Figure A-1 provides the information needed to begin building the appropriate knowledge bases. The condition analysis knowledge base is given in Appendix B. The actions determination knowledge base is given in Appendix C. An example result is given in Appendix D.

In the scenario, the aircraft goes through three flight phases: takeoff, cruise, and landing. A fuel warning is delivered at the 50% mark.

			Overall	Value Range During Flight Phases		
Sensor	Units	Name	Value Range	Takeoff	Cruise	Landing
1	minutes	clock	0, infin	unknown	unknown	unknown
2	degrees	gear position	90, -90	-90	90	-90
3	none	gear lock	0 (unlocked)			
			1 (locked)	1	1	1
4	pounds	fuel	0, 80	>13	>12	>4
5	rpm	engine_1	500, 2600	500,1000	500,1500	1000,2600
6	rpm	engine_2	500, 2600	500,1000	500,1500	1000,2600
7	deg F	engine_1	1, 200	50,150	100,200	1,200
8	deg F	engine_2	1, 200	50,150	100,200	1,200
9	ft/min	vertical speed	0, 1500	750,1500	0,800	500,1000
10	degrees	wing_1 flap	90, -90	>0	~=0	<0
11	degrees	wing_1 flap	90, -90	>0	~=0	<0
12	psi	vacuum	0, 6	4,6	4,6	4,6
13	none	flight phase	1,3	1	2	3

Figure A-1. Example aircraft data sources

APPENDIX B

AN EXAMPLE AIRCRAFT CONDITIONS KNOWLEDGE BASE

```
---------------------------------------------------------------------------
|                        | |EVENT TIMING     |EVENT SPECIFICATION SETS     |
|STATE NAME              |\|DWELL|TRANSITION||S #|VALUE|COP||S #|VALUE|COP|
|------------------------|-|-----|----------||---|-----|---||---|-----|---|

*** Description of flight safety example knowledge base ***

*** Note that the system clock is always sensor #1 (S # 1)

*** Value Range Limits ***

Gear.Lock.Limit.Hi         0     0           2   1     >
Gear.Lock.Limit.Lo         0     0           2   0     <
Gear.Lock.Limit.Hi       \ 0     0           2   1     <=
Gear.Lock.Limit.Lo       \ 0     0           2   0     >=

Gear.Position.Limit.Hi     0     0           3   0     >
Gear.Position.Limit.Lo     0     0           3   -90   <
Gear.Position.Limit.Hi   \ 0     0           3   0     <=
Gear.Position.Limit.Lo   \ 0     0           3   -90   >=

Fuel.Limit.Hi              0     0           4   140   >
Fuel.Limit.Lo              0     0           4   0     <=
Fuel.Limit.Hi            \ 0     0           4   140   <=
Fuel.Limit.Lo            \ 0     0           4   0     >

Engine.1.RPM.Limit.Hi      0     0           5   2600  >
Engine.1.RPM.Limit.Lo      0     0           5   500   <
Engine.1.RPM.Limit.Hi    \ 0     0           5   2600  <=
Engine.1.RPM.Limit.Lo    \ 0     0           5   500   >=
Engine.2.RPM.Limit.Hi          0      0          6    2600   >
Engine.2.RPM.Limit.Lo          0      0          6    500    <
Engine.2.RPM.Limit.Hi        \ 0      0          6    2600   <=
Engine.2.RPM.Limit.Lo        \ 0      0          6    500    >=

Engine.1.Temp.Limit.Hi         0      0          7    200    >
Engine.1.Temp.Limit.Lo         0      0          7    1      <
Engine.1.Temp.Limit.Hi       \ 0      0          7    200    <=
Engine.1.Temp.Limit.Lo       \ 0      0          7    1      >=

Engine.2.Temp.Limit.Hi         0      0          8    200    >
Engine.2.Temp.Limit.Lo         0      0          8    1      <
Engine.2.Temp.Limit.Hi       \ 0      0          8    200    <=
Engine.2.Temp.Limit.Lo       \ 0      0          8    1      >=

Verticle.Speed.Limit.Hi        0      0          9    1500   >
Verticle.Speed.Limit.Lo        0      0          9    0      <
Verticle.Speed.Limit.Hi      \ 0      0          9    1500   <=
Verticle.Speed.Limit.Lo      \ 0      0          9    0      >=

Wing.Flap.1.Limit.Hi           0      0          10   90     >
Wing.Flap.1.Limit.Hi         \ 0      0          10   90     <=
Wing.Flap.1.Limit.Lo           0      0          10   -90    <
Wing.Flap.1.Limit.Lo         \ 0      0          10   -90    >

Wing.Flap.2.Limit.Hi           0      0          11   90     >
Wing.Flap.2.Limit.Hi         \ 0      0          11   90     <=
Wing.Flap.2.Limit.Lo           0      0          11   -90    <
Wing.Flap.2.Limit.Lo         \ 0      0          11   -90    >

Vacuum.Pressure.Limit.Hi       0      0          12   6      >
Vacuum.Pressure.Limit.Lo       0      0          12   4      <
Vacuum.Pressure.Limit.Hi     \ 0      0          12   6      <=
Vacuum.Pressure.Limit.Lo     \ 0      0          12   4      >=
Flight.Phase.Limit.Hi       0     0        13   3     >
Flight.Phase.Limit.Lo       0     0        13   1     <
Flight.Phase.Limit.Hi     \ 0     0        13   3     <=
Flight.Phase.Limit.Lo     \ 0     0        13   1     >=

*** Functional Defs ***

*** Gear Specs

Gear.Down                   10    0        2    1     =     3    -90   =
Gear.Down                 \ 10    0        2    1     <>
Gear.Down                 \ 10    0        3    -90   <>

Gear.Up                     10    0        2    1     =     3    0     =
Gear.Up                   \ 10    0        2    1     <>
Gear.Up                   \ 10    0        3    0     <>

*** Fuel Specs

Take.Off.Fuel               10    0        4    13    >
Take.Off.Fuel             \ 10    0        4    13    <=

Cruise.Fuel                 10    0        4    12    >
Cruise.Fuel               \ 10    0        4    12    <=

Landing.Fuel                10    0        4    4     >
Landing.Fuel              \ 10    0        4    4     <=

Fuel.Warning                10    0        4    70    <=
Fuel.Warning              \ 10    0        4    70    >

*** Engine 1 Specs

Take.Off.Engine.1.RPM       10    0        5    500   >=    5    1000  <=
Take.Off.Engine.1.RPM     \ 10    0        5    500   <
Take.Off.Engine.1.RPM     \ 10    0        5    1000  >
```

```
Cruise.Engine.1.RPM          10    0          5   500    >=     5   1500   <=
Cruise.Engine.1.RPM        \ 10    0          5   500    <
Cruise.Engine.1.RPM        \ 10    0          5   1500   >

Landing.Engine.1.RPM         10    0          5   1000   >=     5   2600   <=
Landing.Engine.1.RPM       \ 10    0          5   1000   <
Landing.Engine.1.RPM       \ 10    0          5   2600   >

Take.Off.Engine.1.Temp       10    0          7   50     >=     7   150    <=
Take.Off.Engine.1.Temp     \ 10    0          7   50     <
Take.Off.Engine.1.Temp     \ 10    0          7   150    >

Cruise.Engine.1.Temp         10    0          7   100    >=     7   200    <=
Cruise.Engine.1.Temp       \ 10    0          7   100    <
Cruise.Engine.1.Temp       \ 10    0          7   200    >

Landing.Engine.1.Temp        10    0          7   1      >=     7   200    <=
Landing.Engine.1.Temp      \ 10    0          7   1      <
Landing.Engine.1.Temp      \ 10    0          7   200    >

*** Engine 2 Specs ***

Take.Off.Engine.2.RPM        10    0          6   500    >=     6   1000   <=
Take.Off.Engine.2.RPM      \ 10    0          6   500    <
Take.Off.Engine.2.RPM      \ 10    0          6   1000   >

Cruise.Engine.2.RPM          10    0          6   500    >=     6   1500   <=
Cruise.Engine.2.RPM        \ 10    0          6   500    <
Cruise.Engine.2.RPM        \ 10    0          6   1500   >

Landing.Engine.2.RPM         10    0          6   1000   >=     6   2600   <=
Landing.Engine.2.RPM       \ 10    0          6   1000   <
Landing.Engine.2.RPM       \ 10    0          6   2600   >

Take.Off.Engine.2.Temp       10    0          8   50     >=     8   150    <=
Take.Off.Engine.2.Temp     \ 10    0          8   50     <
Take.Off.Engine.2.Temp     \ 10    0          8   150    >
```

```
Cruise.Engine.2.Temp        10    0        8   100   >=   8   200   <=
Cruise.Engine.2.Temp      \ 10    0        8   100   <
Cruise.Engine.2.Temp      \ 10    0        8   200   >

Landing.Engine.2.Temp       10    0        8   1     >=   8   200   <=
Landing.Engine.2.Temp     \ 10    0        8   1     <
Landing.Engine.2.Temp     \ 10    0        8   200   >

*** Vertical Speed Specs

Take.Off.Vertical.Speed     10    0        9   750   >=   9   1500  <=
Take.Off.Vertical.Speed   \ 10    0        9   750   <
Take.Off.Vertical.Speed   \ 10    0        9   1500  >

Cruise.Vertical.Speed       10    0        9   0     >=   9   800   <=
Cruise.Vertical.Speed     \ 10    0        9   0     <
Cruise.Vertical.Speed     \ 10    0        9   800   >

Landing.Vertical.Speed      10    0        9   0     >=   9   1500  <=
Landing.Vertical.Speed    \ 10    0        9   0     <
Landing.Vertical.Speed    \ 10    0        9   1500  >

*** Wing Flap 1 Specs

Take.Off.Wing.Flap.1        10    0        10  0     >=   10  90    <=
Take.Off.Wing.Flap.1      \ 10    0        10  0     <
Take.Off.Wing.Flap.1      \ 10    0        10  90    >=

Cruise.Wing.Flap.1          10    0        10  10    <=   10  -10   >=
Cruise.Wing.Flap.1        \ 10    0        10  10    >
Cruise.Wing.Flap.1        \ 10    0        10  -10   <

Landing.Wing.Flap.1         10    0        10  -90   >=   10  0     <=
Landing.Wing.Flap.1       \ 10    0        10  -90   <
Landing.Wing.Flap.1       \ 10    0        10  0     >
```

```
*** Wing Flap 2 Specs

Take.Off.Wing.Flap.2          10     0          11  0      >=    11  90     <=
Take.Off.Wing.Flap.2        \ 10     0          11  0      <
Take.Off.Wing.Flap.2        \ 10     0          11  90     >=

Cruise.Wing.Flap.2            10     0          11  10     <=    11  -10    >=
Cruise.Wing.Flap.2          \ 10     0          11  10     >
Cruise.Wing.Flap.2          \ 10     0          11  -10    <

Landing.Wing.Flap.2           10     0          11  -90    >=    11  0      <=
Landing.Wing.Flap.2         \ 10     0          11  -90    <
Landing.Wing.Flap.2         \ 10     0          11  0      >

*** Vacuum Pressure Specs

Take.Off.Vacuum.Pressure      10     0          12  4      >=    12  6      <=
Take.Off.Vacuum.Pressure    \ 10     0          12  4      <
Take.Off.Vacuum.Pressure    \ 10     0          12  6      >

Cruise.Vacuum.Pressure        10     0          12  4      >=    12  6      <=
Cruise.Vacuum.Pressure      \ 10     0          12  4      <
Cruise.Vacuum.Pressure      \ 10     0          12  6      >

Landing.Vacuum.Pressure       10     0          12  4      >=    12  6      <=
Landing.Vacuum.Pressure     \ 10     0          12  4      <
Landing.Vacuum.Pressure     \ 10     0          12  6      >

*** Flight Phase Specs

Take.Off.Flight.Phase         10     0          13  1      =
Take.Off.Flight.Phase       \ 10     0          13  1      <>

Cruise.Flight.Phase           10     0          13  2      =
Cruise.Flight.Phase         \ 10     0          13  2      <>
```

```
Landing.Flight.Phase          10     0          13   3      =
Landing.Flight.Phase        \ 10     0          13   3      <>

*** END ***
```

APPENDIX C

AN EXAMPLE DECISIONS/CRITERIA KNOWLEDGE BASE

```
Ready for taxi
taxi
no_taxi
Gear.Down
Take.Off.Fuel
Take.Off.Engine.1.Temp
Take.Off.Engine.2.Temp
\
Ready for takeoff
takeoff
no_take
Gear.Down
Take.Off.Fuel
Take.Off.Engine.1.RPM
Take.Off.Engine.2.RPM
Take.Off.Engine.1.Temp
Take.Off.Engine.2.Temp
Take.Off.Vacuum.Pressure
\
Ready for cruising
cruise
no_cruis
Gear.Up
Cruise.Fuel
Cruise.Engine.1.RPM
Cruise.Engine.2.RPM
Cruise.Engine.1.Temp
Cruise.Engine.2.Temp
Cruise.Vertical.Speed
Cruise.Wing.Flap.1
Cruise.Wing.Flap.2
\
Ready for landing
land
no_land
Gear.Down
Landing.Engine.1.RPM
Landing.Engine.2.RPM
Landing.Engine.1.Temp
Landing.Engine.2.Temp
Landing.Vertical.Speed
Landing.Wing.Flap.1
Landing.Wing.Flap.2
Landing.Vacuum.Pressure
\
Fuel Warning #1
fuel_1
go_fuel
Fuel.Warning
\
Fuel Warning #2
Fuel_2
go_fuel
~Landing.Fuel
\
```

APPENDIX D

AN EXAMPLE OF RUNNING THE ENTIRE SYSTEM

Data Sources and Their Values Over Time

Time	1	2	3	4	5	6	7	8	9	10	11	12
0	1	-90	140	0	0	0	0	0	10	10	0	0
1	1	-90	140	0	0	0	0	0	10	10	0	0
2	1	-90	140	0	0	0	0	0	10	10	0	1
3	1	-90	140	0	0	0	0	0	10	10	0	1
4	1	-90	140	0	0	0	0	0	10	10	0	1
5	1	-90	140	0	0	0	0	0	10	10	0	1
6	1	-90	140	0	0	0	0	0	10	10	0	1
7	1	-90	140	0	0	0	0	0	10	10	0	1
8	1	-90	140	0	0	0	0	0	10	10	0	1
9	1	-90	140	0	0	0	0	0	10	10	0	1
10	1	-90	140	0	0	0	0	0	10	10	0	1
11	1	-90	140	0	0	0	0	0	10	10	0	1
12	1	-90	140	0	0	0	0	0	10	10	0	1
13	1	-90	140	0	0	0	0	0	10	10	0	1
14	1	-90	140	0	0	0	0	0	10	10	0	1
15	1	-90	140	300	300	100	100	0	10	10	3	1
16	1	-90	140	300	300	100	100	0	10	10	3	1
17	1	-90	140	300	300	100	100	0	10	10	3	1
18	1	-90	140	300	300	100	100	0	10	10	3	1
19	1	-90	140	300	300	100	100	0	10	10	3	1
20	1	-90	140	300	300	100	100	0	10	10	3	1
21	1	-90	140	300	300	100	100	0	10	10	3	1
22	1	-90	140	300	300	100	100	0	10	10	3	1
23	1	-90	140	300	300	100	100	0	10	10	3	1
24	1	-90	140	300	300	100	100	0	10	10	3	1
25	1	-90	140	500	500	100	100	0	10	10	4	1
26	1	-90	140	500	500	100	100	0	10	10	4	1
27	1	-90	140	500	500	100	100	0	10	10	4	1
28	1	-90	140	500	500	100	100	0	10	10	4	1
29	1	-90	140	500	500	100	100	0	10	10	4	1
30	1	-90	130	500	500	100	100	0	10	10	4	1
31	1	-90	130	500	500	100	100	0	10	10	4	1
32	1	-90	130	500	500	100	100	0	10	10	4	1
33	1	-90	130	500	500	100	100	0	10	10	4	1
34	1	-90	130	500	500	100	100	0	10	10	4	1
35	1	-90	130	500	500	100	100	0	10	10	4	1
36	1	-90	130	500	500	100	100	0	10	10	4	1
37	1	-90	130	500	500	100	100	0	10	10	4	1
38	1	-90	130	500	500	100	100	0	10	10	4	1
39	1	-90	130	500	500	100	100	0	10	10	4	1
40	1	-90	130	500	500	200	200	0	10	10	5	1
41	1	-90	130	500	500	200	200	0	10	10	5	1
42	1	-90	130	500	500	200	200	0	10	10	5	1

43	1	-90	130	500	500	200	200	0	10	10	5	1
44	1	-90	130	500	500	200	200	0	10	10	5	1
45	1	-90	130	500	500	200	200	0	10	10	5	1
46	1	-90	130	500	500	200	200	0	10	10	5	1
47	1	-90	130	500	500	200	200	0	10	10	5	1
48	1	-90	130	500	500	200	200	0	10	10	5	1
49	1	-90	130	500	500	200	200	0	10	10	5	1
50	1	-90	130	750	750	200	200	0	10	10	6	1
51	1	-90	130	750	750	200	200	0	10	10	6	1
52	1	-90	130	750	750	200	200	0	10	10	6	1
53	1	-90	130	750	750	200	200	0	10	10	6	1
54	1	-90	130	750	750	200	200	0	10	10	6	1
55	1	-90	130	750	750	200	200	0	10	10	6	1
56	1	-90	130	750	750	200	200	0	10	10	6	1
57	1	-90	130	750	750	200	200	0	10	10	6	1
58	1	-90	130	750	750	200	200	0	10	10	6	1
59	1	-90	130	750	750	200	200	0	10	10	6	1
60	1	-90	120	750	750	200	200	0	10	10	6	1
61	1	-90	120	750	750	200	200	0	10	10	6	1
62	1	-90	120	750	750	200	200	0	10	10	6	1
63	1	-90	120	750	750	200	200	0	10	10	6	1
64	1	-90	120	750	750	200	200	0	10	10	6	1
65	1	-90	120	750	750	200	200	0	10	10	6	1
66	1	-90	120	750	750	200	200	0	10	10	6	1
67	1	-90	120	750	750	200	200	0	10	10	6	1
68	1	-90	120	750	750	200	200	0	10	10	6	1
69	1	-90	120	750	750	200	200	0	10	10	6	1
70	1	-90	120	750	750	200	200	0	10	10	6	1
71	1	-90	120	750	750	200	200	0	10	10	6	1
72	1	-90	120	750	750	200	200	0	10	10	6	1
73	1	-90	120	750	750	200	200	0	10	10	6	1
74	1	-90	120	750	750	200	200	0	10	10	6	1
75	1	-90	120	750	750	200	200	0	50	50	6	1
76	1	-90	120	750	750	200	200	0	50	50	6	1
77	1	-90	120	750	750	200	200	0	50	50	6	1
78	1	-90	120	750	750	200	200	0	50	50	6	1
79	1	-90	120	750	750	200	200	0	50	50	6	1
80	1	-90	120	750	750	200	200	0	50	50	6	1
81	1	-90	120	750	750	200	200	0	50	50	6	1
82	1	-90	120	750	750	200	200	0	50	50	6	1
83	1	-90	120	750	750	200	200	0	50	50	6	1
84	1	-90	120	750	750	200	200	0	50	50	6	1
85	1	-90	120	750	750	200	200	0	50	50	6	1
86	1	-90	120	750	750	200	200	0	50	50	6	1
87	1	-90	120	750	750	200	200	0	50	50	6	1
88	1	-90	120	750	750	200	200	0	50	50	6	1
89	1	-90	120	750	750	200	200	0	50	50	6	1
90	1	-90	110	750	750	200	200	0	50	50	6	1
91	1	-90	110	750	750	200	200	0	50	50	6	1
92	1	-90	110	750	750	200	200	0	50	50	6	1
93	1	-90	110	750	750	200	200	0	50	50	6	1
94	1	-90	110	750	750	200	200	0	50	50	6	1

95	1	-90	110	750	750	200	200	0	50	50	6	1
96	1	-90	110	750	750	200	200	0	50	50	6	1
97	1	-90	110	750	750	200	200	0	50	50	6	1
98	1	-90	110	750	750	200	200	0	50	50	6	1
99	1	-90	110	750	750	200	200	0	50	50	6	1
100	1	-90	110	1000	1000	200	200	300	50	50	6	1
101	1	-90	110	1000	1000	200	200	300	50	50	6	1
102	1	-90	110	1000	1000	200	200	300	50	50	6	1
103	1	-90	110	1000	1000	200	200	300	50	50	6	1
104	1	-90	110	1000	1000	200	200	300	50	50	6	1
105	1	-90	110	1000	1000	200	200	300	50	50	6	1
106	1	-90	110	1000	1000	200	200	300	50	50	6	1
107	1	-90	110	1000	1000	200	200	300	50	50	6	1
108	1	-90	110	1000	1000	200	200	300	50	50	6	1
109	1	-90	110	1000	1000	200	200	300	50	50	6	1
110	1	-90	110	1000	1000	200	200	300	50	50	6	1
111	1	-90	110	1000	1000	200	200	300	50	50	6	1
112	1	-90	110	1000	1000	200	200	300	50	50	6	1
113	1	-90	110	1000	1000	200	200	300	50	50	6	1
114	1	-90	110	1000	1000	200	200	300	50	50	6	1
115	1	-90	110	1000	1000	200	200	300	50	50	6	1
116	1	-90	110	1000	1000	200	200	300	50	50	6	1
117	1	-90	110	1000	1000	200	200	300	50	50	6	1
118	1	-90	110	1000	1000	200	200	300	50	50	6	1
119	1	-90	110	1000	1000	200	200	300	50	50	6	1
120	1	-90	100	1000	1000	200	200	300	50	50	6	1
121	1	-90	100	1000	1000	200	200	300	50	50	6	1
122	1	-90	100	1000	1000	200	200	300	50	50	6	1
123	1	-90	100	1000	1000	200	200	300	50	50	6	1
124	1	-90	100	1000	1000	200	200	300	50	50	6	1
125	1	-90	100	1000	1000	200	200	300	50	50	6	1
126	1	-90	100	1000	1000	200	200	300	50	50	6	1
127	1	-90	100	1000	1000	200	200	300	50	50	6	1
128	1	-90	100	1000	1000	200	200	300	50	50	6	1
129	1	-90	100	1000	1000	200	200	300	50	50	6	1
130	1	-90	100	1000	1000	200	200	300	50	50	6	1
131	1	-90	100	1000	1000	200	200	300	50	50	6	1
132	1	-90	100	1000	1000	200	200	300	50	50	6	1
133	1	-90	100	1000	1000	200	200	300	50	50	6	1
134	1	-90	100	1000	1000	200	200	300	50	50	6	1
135	1	-90	100	1000	1000	200	200	300	50	50	6	1
136	1	-90	100	1000	1000	200	200	300	50	50	6	1
137	1	-90	100	1000	1000	200	200	300	50	50	6	1
138	1	-90	100	1000	1000	200	200	300	50	50	6	1
139	1	-90	100	1000	1000	200	200	300	50	50	6	1
140	1	-90	100	1000	1000	200	200	300	50	50	6	1
141	1	-90	100	1000	1000	200	200	300	50	50	6	1
142	1	-90	100	1000	1000	200	200	300	50	50	6	1
143	1	-90	100	1000	1000	200	200	300	50	50	6	1
144	1	-90	100	1000	1000	200	200	300	50	50	6	1
145	1	-90	100	1000	1000	200	200	300	50	50	6	1
146	1	-90	100	1000	1000	200	200	300	50	50	6	1

147	1	-90	100	1000	1000	200	200	300	50	50	6	1
148	1	-90	100	1000	1000	200	200	300	50	50	6	1
149	1	-90	100	1000	1000	200	200	300	50	50	6	1
150	1	-90	90	1000	1000	200	200	300	50	50	6	1
151	1	-90	90	1000	1000	200	200	300	50	50	6	1
152	1	-90	90	1000	1000	200	200	300	50	50	6	1
153	1	-90	90	1000	1000	200	200	300	50	50	6	1
154	1	-90	90	1000	1000	200	200	300	50	50	6	1
155	1	-90	90	1000	1000	200	200	300	50	50	6	1
156	1	-90	90	1000	1000	200	200	300	50	50	6	1
157	1	-90	90	1000	1000	200	200	300	50	50	6	1
158	1	-90	90	1000	1000	200	200	300	50	50	6	1
159	1	-90	90	1000	1000	200	200	300	50	50	6	1
160	1	-90	90	1000	1000	200	200	300	50	50	6	1
161	1	-90	90	1000	1000	200	200	300	50	50	6	1
162	1	-90	90	1000	1000	200	200	300	50	50	6	1
163	1	-90	90	1000	1000	200	200	300	50	50	6	1
164	1	-90	90	1000	1000	200	200	300	50	50	6	1
165	1	-90	90	1000	1000	200	200	300	50	50	6	1
166	1	-90	90	1000	1000	200	200	300	50	50	6	1
167	1	-90	90	1000	1000	200	200	300	50	50	6	1
168	1	-90	90	1000	1000	200	200	300	50	50	6	1
169	1	-90	90	1000	1000	200	200	300	50	50	6	1
170	1	-90	90	1000	1000	200	200	300	50	50	6	1
171	1	-90	90	1000	1000	200	200	300	50	50	6	1
172	1	-90	90	1000	1000	200	200	300	50	50	6	1
173	1	-90	90	1000	1000	200	200	300	50	50	6	1
174	1	-90	90	1000	1000	200	200	300	50	50	6	1
175	1	-90	90	1000	1000	200	200	300	50	50	6	1
176	1	-90	90	1000	1000	200	200	300	50	50	6	1
177	1	-90	90	1000	1000	200	200	300	50	50	6	1
178	1	-90	90	1000	1000	200	200	300	50	50	6	1
179	1	-90	90	1000	1000	200	200	300	50	50	6	1
180	1	-90	80	1000	1000	200	200	300	50	50	6	1
181	1	-90	80	1000	1000	200	200	300	50	50	6	1
182	1	-90	80	1000	1000	200	200	300	50	50	6	1
183	1	-90	80	1000	1000	200	200	300	50	50	6	1
184	1	-90	80	1000	1000	200	200	300	50	50	6	1
185	1	-90	80	1000	1000	200	200	300	50	50	6	1
186	1	-90	80	1000	1000	200	200	300	50	50	6	1
187	1	-90	80	1000	1000	200	200	300	50	50	6	1
188	1	-90	80	1000	1000	200	200	300	50	50	6	1
189	1	-90	80	1000	1000	200	200	300	50	50	6	1
190	1	-90	80	1000	1000	200	200	300	50	50	6	1
191	1	-90	80	1000	1000	200	200	300	50	50	6	1
192	1	-90	80	1000	1000	200	200	300	50	50	6	1
193	1	-90	80	1000	1000	200	200	300	50	50	6	1
194	1	-90	80	1000	1000	200	200	300	50	50	6	1
195	1	-90	80	1000	1000	200	200	300	50	50	6	1
196	1	-90	80	1000	1000	200	200	300	50	50	6	1
197	1	-90	80	1000	1000	200	200	300	50	50	6	1
198	1	-90	80	1000	1000	200	200	300	50	50	6	1

199	1	-90	80	1000	1000	200	200	300	50	50	6	1
200	1	-90	80	1000	1000	200	200	750	50	50	6	1
201	1	-90	80	1000	1000	200	200	750	50	50	6	1
202	1	-90	80	1000	1000	200	200	750	50	50	6	1
203	1	-90	80	1000	1000	200	200	750	50	50	6	1
204	1	-90	80	1000	1000	200	200	750	50	50	6	1
205	1	-90	80	1000	1000	200	200	750	80	80	6	1
206	1	-90	80	1000	1000	200	200	750	80	80	6	1
207	1	-90	80	1000	1000	200	200	750	80	80	6	1
208	1	-90	80	1000	1000	200	200	750	80	80	6	1
209	1	-90	80	1000	1000	200	200	750	80	80	6	1
210	1	-90	70	1000	1000	200	200	750	80	80	6	1
211	1	-90	70	1000	1000	200	200	750	80	80	6	1
212	1	-90	70	1000	1000	200	200	750	80	80	6	1
213	1	-90	70	1000	1000	200	200	750	80	80	6	1
214	1	-90	70	1000	1000	200	200	750	80	80	6	1
215	1	-90	70	1000	1000	200	200	750	80	80	6	1
216	1	-90	70	1000	1000	200	200	750	80	80	6	1
217	1	-90	70	1000	1000	200	200	750	80	80	6	1
218	1	-90	70	1000	1000	200	200	750	80	80	6	1
219	1	-90	70	1000	1000	200	200	750	80	80	6	1
220	1	-90	70	1000	1000	200	200	750	80	80	6	1
221	1	-90	70	1000	1000	200	200	750	80	80	6	1
222	1	-90	70	1000	1000	200	200	750	80	80	6	1
223	1	-90	70	1000	1000	200	200	750	80	80	6	1
224	1	-90	70	1000	1000	200	200	750	80	80	6	1
225	1	0	70	1000	1000	200	200	750	80	80	6	1
226	1	0	70	1000	1000	200	200	750	80	80	6	1
227	1	0	70	1000	1000	200	200	750	80	80	6	1
228	1	0	70	1000	1000	200	200	750	80	80	6	1
229	1	0	70	1000	1000	200	200	750	80	80	6	1
230	1	0	70	1000	1000	200	200	750	80	80	6	1
231	1	0	70	1000	1000	200	200	750	80	80	6	1
232	1	0	70	1000	1000	200	200	750	80	80	6	1
233	1	0	70	1000	1000	200	200	750	80	80	6	1
234	1	0	70	1000	1000	200	200	750	80	80	6	1
235	1	0	70	1000	1000	200	200	750	80	80	6	1
236	1	0	70	1000	1000	200	200	750	80	80	6	1
237	1	0	70	1000	1000	200	200	750	80	80	6	1
238	1	0	70	1000	1000	200	200	750	80	80	6	1
239	1	0	70	1000	1000	200	200	750	80	80	6	1
240	1	0	60	1000	1000	200	200	750	80	80	6	1
241	1	0	60	1000	1000	200	200	750	80	80	6	1
242	1	0	60	1000	1000	200	200	750	80	80	6	1
243	1	0	60	1000	1000	200	200	750	80	80	6	1
244	1	0	60	1000	1000	200	200	750	80	80	6	1
245	1	0	60	1000	1000	200	200	750	80	80	6	1
246	1	0	60	1000	1000	200	200	750	80	80	6	1
247	1	0	60	1000	1000	200	200	750	80	80	6	1
248	1	0	60	1000	1000	200	200	750	80	80	6	1
249	1	0	60	1000	1000	200	200	750	80	80	6	1
250	1	0	60	1000	1000	200	200	750	5	5	6	1

251	1	0	60	1000	1000	200	200	750	5	5	6	1
252	1	0	60	1000	1000	200	200	750	5	5	6	1
253	1	0	60	1000	1000	200	200	750	5	5	6	1
254	1	0	60	1000	1000	200	200	750	5	5	6	1
255	1	0	60	1000	1000	200	200	750	5	5	6	1
256	1	0	60	1000	1000	200	200	750	5	5	6	1
257	1	0	60	1000	1000	200	200	750	5	5	6	1
258	1	0	60	1000	1000	200	200	750	5	5	6	1
259	1	0	60	1000	1000	200	200	750	5	5	6	1
260	1	0	60	1000	1000	200	200	750	5	5	6	1
261	1	0	60	1000	1000	200	200	750	5	5	6	1
262	1	0	60	1000	1000	200	200	750	5	5	6	1
263	1	0	60	1000	1000	200	200	750	5	5	6	1
264	1	0	60	1000	1000	200	200	750	5	5	6	1
265	1	0	60	1000	1000	200	200	750	5	5	6	1
266	1	0	60	1000	1000	200	200	750	5	5	6	1
267	1	0	60	1000	1000	200	200	750	5	5	6	1
268	1	0	60	1000	1000	200	200	750	5	5	6	1
269	1	0	60	1000	1000	200	200	750	5	5	6	1
270	1	0	50	1000	1000	200	200	750	5	5	6	1
271	1	0	50	1000	1000	200	200	750	5	5	6	1
272	1	0	50	1000	1000	200	200	750	5	5	6	1
273	1	0	50	1000	1000	200	200	750	5	5	6	1
274	1	0	50	1000	1000	200	200	750	5	5	6	1
275	1	0	50	1000	1000	200	200	750	5	5	6	1
276	1	0	50	1000	1000	200	200	750	5	5	6	1
277	1	0	50	1000	1000	200	200	750	5	5	6	1
278	1	0	50	1000	1000	200	200	750	5	5	6	1
279	1	0	50	1000	1000	200	200	750	5	5	6	1
280	1	0	50	1000	1000	200	200	750	5	5	6	1
281	1	0	50	1000	1000	200	200	750	5	5	6	1
282	1	0	50	1000	1000	200	200	750	5	5	6	1
283	1	0	50	1000	1000	200	200	750	5	5	6	1
284	1	0	50	1000	1000	200	200	750	5	5	6	1
285	1	0	50	1000	1000	200	200	750	5	5	6	1
286	1	0	50	1000	1000	200	200	750	5	5	6	1
287	1	0	50	1000	1000	200	200	750	5	5	6	1
288	1	0	50	1000	1000	200	200	750	5	5	6	1
289	1	0	50	1000	1000	200	200	750	5	5	6	1
290	1	0	50	1000	1000	200	200	750	5	5	6	1
291	1	0	50	1000	1000	200	200	750	5	5	6	1
292	1	0	50	1000	1000	200	200	750	5	5	6	1
293	1	0	50	1000	1000	200	200	750	5	5	6	1
294	1	0	50	1000	1000	200	200	750	5	5	6	1
295	1	0	50	1000	1000	200	200	750	5	5	6	1
296	1	0	50	1000	1000	200	200	750	5	5	6	1
297	1	0	50	1000	1000	200	200	750	5	5	6	1
298	1	0	50	1000	1000	200	200	750	5	5	6	1
299	1	0	50	1000	1000	200	200	750	5	5	6	1
300	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
301	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
302	1	0	40	1000	1000	200	200	1500	-50	-50	6	1

303	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
304	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
305	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
306	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
307	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
308	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
309	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
310	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
311	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
312	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
313	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
314	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
315	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
316	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
317	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
318	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
319	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
320	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
321	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
322	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
323	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
324	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
325	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
326	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
327	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
328	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
329	1	0	40	1000	1000	200	200	1500	-50	-50	6	1
330	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
331	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
332	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
333	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
334	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
335	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
336	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
337	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
338	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
339	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
340	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
341	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
342	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
343	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
344	1	0	30	1000	1000	200	200	1500	-50	-50	6	1
345	0	0	30	1000	1000	200	200	1500	-50	-50	6	1
346	0	0	30	1000	1000	200	200	1500	-50	-50	6	1
347	0	0	30	1000	1000	200	200	1500	-50	-50	6	1
348	0	0	30	1000	1000	200	200	1500	-50	-50	6	1
349	0	0	30	1000	1000	200	200	1500	-50	-50	6	1
350	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
351	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
352	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
353	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
354	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1

355	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
356	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
357	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
358	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
359	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
360	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
361	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
362	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
363	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
364	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
365	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
366	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
367	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
368	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
369	1	-90	30	1000	1000	200	200	1500	-50	-50	6	1
370	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
371	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
372	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
373	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
374	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
375	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
376	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
377	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
378	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
379	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
380	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
381	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
382	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
383	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
384	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
385	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
386	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
387	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
388	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
389	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
390	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
391	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
392	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
393	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
394	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
395	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
396	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
397	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
398	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
399	1	-90	20	1000	1000	200	200	1500	-50	-50	6	1
400	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
401	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
402	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
403	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
404	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
405	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
406	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1

407	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
408	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
409	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
410	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
411	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
412	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
413	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
414	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
415	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
416	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
417	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
418	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
419	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
420	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
421	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
422	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
423	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
424	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
425	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
426	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
427	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
428	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
429	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
430	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
431	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
432	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
433	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
434	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
435	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
436	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
437	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
438	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
439	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
440	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
441	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
442	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
443	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
444	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
445	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
446	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
447	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
448	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
449	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1
450	1	-90	10	1000	1000	200	200	1500	-50	-50	6	1

Result of Running the Decision Support System

```
READING AIRCRAFT CONDITION: Engine.1.RPM.Limit.Lo AT TIME = 0
READING AIRCRAFT CONDITION: Engine.2.RPM.Limit.Lo AT TIME = 0
READING AIRCRAFT CONDITION: Engine.1.Temp.Limit.Lo AT TIME = 0
READING AIRCRAFT CONDITION: Engine.2.Temp.Limit.Lo AT TIME = 0
READING AIRCRAFT CONDITION: Vacuum.Pressure.Limit.Lo AT TIME = 0
READING AIRCRAFT CONDITION: Flight.Phase.Limit.Lo AT TIME = 0
READING AIRCRAFT CONDITION: Flight.Phase.Limit.Lo\ AT TIME = 4
READING AIRCRAFT CONDITION: Gear.Down AT TIME = 12
READING AIRCRAFT CONDITION: Take.Off.Fuel AT TIME = 12
READING AIRCRAFT CONDITION: Cruise.Fuel AT TIME = 12
READING AIRCRAFT CONDITION: Landing.Fuel AT TIME = 12
READING AIRCRAFT CONDITION: Cruise.Vertical.Speed AT TIME = 12
READING AIRCRAFT CONDITION: Landing.Vertical.Speed AT TIME = 12
READING AIRCRAFT CONDITION: Take.Off.Wing.Flap.1 AT TIME = 12
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.1 AT TIME = 12
READING AIRCRAFT CONDITION: Take.Off.Wing.Flap.2 AT TIME = 12
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.2 AT TIME = 12
READING AIRCRAFT CONDITION: Engine.1.Temp.Limit.Lo\ AT TIME = 16
READING AIRCRAFT CONDITION: Engine.2.Temp.Limit.Lo\ AT TIME = 16
READING AIRCRAFT CONDITION: Take.Off.Flight.Phase AT TIME = 16
READING AIRCRAFT CONDITION: Engine.1.RPM.Limit.Lo\ AT TIME = 28
READING AIRCRAFT CONDITION: Engine.2.RPM.Limit.Lo\ AT TIME = 28
READING AIRCRAFT CONDITION: Vacuum.Pressure.Limit.Lo\ AT TIME = 28
READING AIRCRAFT CONDITION: Take.Off.Engine.1.Temp AT TIME = 28
READING AIRCRAFT CONDITION: Cruise.Engine.1.Temp AT TIME = 28
READING AIRCRAFT CONDITION: Landing.Engine.1.Temp AT TIME = 28
READING AIRCRAFT CONDITION: Take.Off.Engine.2.Temp AT TIME = 28

FIRING RULE: Ready for taxi

**** BEGIN TASK: taxi ****

The aircraft is ready for taxi

**** taxi ****

  The following conditions were true at the time the rule fired:
  -> Gear.Down
  -> Take.Off.Fuel
     Cruise.Fuel
     Landing.Fuel
     Cruise.Vertical.Speed
     Landing.Vertical.Speed
     Take.Off.Wing.Flap.1
     Cruise.Wing.Flap.1
     Take.Off.Wing.Flap.2
     Cruise.Wing.Flap.2
     Take.Off.Flight.Phase
  -> Take.Off.Engine.1.Temp
     Cruise.Engine.1.Temp
     Landing.Engine.1.Temp
  -> Take.Off.Engine.2.Temp

READING AIRCRAFT CONDITION: Cruise.Engine.2.Temp AT TIME = 28
READING AIRCRAFT CONDITION: Landing.Engine.2.Temp AT TIME = 28
READING AIRCRAFT CONDITION: Take.Off.Engine.1.RPM AT TIME = 40
READING AIRCRAFT CONDITION: Cruise.Engine.1.RPM AT TIME = 40
READING AIRCRAFT CONDITION: Take.Off.Engine.2.RPM AT TIME = 40
READING AIRCRAFT CONDITION: Cruise.Engine.2.RPM AT TIME = 40
READING AIRCRAFT CONDITION: Take.Off.Vacuum.Pressure AT TIME = 40

FIRING RULE: Ready for takeoff

**** BEGIN TASK: takeoff ****

The aircraft is ready for takeoff

**** takeoff ****

  The following conditions were true at the time the rule fired:
  -> Gear.Down
  -> Take.Off.Fuel
     Cruise.Fuel
     Landing.Fuel
     Cruise.Vertical.Speed
     Landing.Vertical.Speed
     Take.Off.Wing.Flap.1
     Cruise.Wing.Flap.1
     Take.Off.Wing.Flap.2
     Cruise.Wing.Flap.2
     Take.Off.Flight.Phase
  -> Take.Off.Engine.1.Temp
     Cruise.Engine.1.Temp
     Landing.Engine.1.Temp
  -> Take.Off.Engine.2.Temp
     Cruise.Engine.2.Temp
     Landing.Engine.2.Temp
  -> Take.Off.Engine.1.RPM
     Cruise.Engine.1.RPM
  -> Take.Off.Engine.2.RPM
     Cruise.Engine.2.RPM
  -> Take.Off.Vacuum.Pressure

READING AIRCRAFT CONDITION: Cruise.Vacuum.Pressure AT TIME = 40
READING AIRCRAFT CONDITION: Landing.Vacuum.Pressure AT TIME = 40
READING AIRCRAFT CONDITION: Take.Off.Engine.1.Temp\ AT TIME = 52

DEACTIVATING RULE'S TASK EXECUTION LIST: Ready for taxi
```

```
**** BEGIN TASK: no_taxi ****

The aircraft is no longer in taxi mode

**** no_taxi ****

REMOVING TASK taxi FROM TASK EXECUTION LIST

DEACTIVATING RULE'S TASK EXECUTION LIST: Ready for takeoff

**** BEGIN TASK: no_take ****

The aircraft is no longer in takeoff mode

**** no_take ****

REMOVING TASK takeoff FROM TASK EXECUTION LIST

READING AIRCRAFT CONDITION: Take.Off.Engine.2.Temp\ AT TIME = 52
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.1\ AT TIME = 88
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.2\ AT TIME = 88
READING AIRCRAFT CONDITION: Landing.Engine.1.RPM AT TIME = 112
READING AIRCRAFT CONDITION: Landing.Engine.2.RPM AT TIME = 112
READING AIRCRAFT CONDITION: Take.Off.Vertical.Speed AT TIME = 212
READING AIRCRAFT CONDITION: Fuel.Warning AT TIME = 224

FIRING RULE: Fuel Warning #1

**** BEGIN TASK: fuel_1 ****

The fuel level is 50% or less down

**** fuel_1 ****

  The following conditions were true at the time the rule fired:
     Gear.Down
     Take.Off.Fuel
     Cruise.Fuel
     Landing.Fuel
     Cruise.Vertical.Speed
     Landing.Vertical.Speed
     Take.Off.Wing.Flap.1
     Take.Off.Wing.Flap.2
     Take.Off.Flight.Phase
     Cruise.Engine.1.Temp
     Landing.Engine.1.Temp
     Cruise.Engine.2.Temp
     Landing.Engine.2.Temp
     Take.Off.Engine.1.RPM
     Cruise.Engine.1.RPM
     Take.Off.Engine.2.RPM
     Cruise.Engine.2.RPM
     Take.Off.Vacuum.Pressure
     Cruise.Vacuum.Pressure
     Landing.Vacuum.Pressure
     Landing.Engine.1.RPM
     Landing.Engine.2.RPM
     Take.Off.Vertical.Speed
  -> Fuel.Warning

READING AIRCRAFT CONDITION: Gear.Down\ AT TIME = 240
READING AIRCRAFT CONDITION: Gear.Up AT TIME = 240
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.1 AT TIME = 264
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.2 AT TIME = 264

FIRING RULE: Ready for cruising

**** BEGIN TASK: cruise ****

The aircraft is in cruise mode

**** cruise ****

  The following conditions were true at the time the rule fired:
     Take.Off.Fuel
  -> Cruise.Fuel
     Landing.Fuel
  -> Cruise.Vertical.Speed
     Landing.Vertical.Speed
     Take.Off.Wing.Flap.1
     Take.Off.Wing.Flap.2
     Take.Off.Flight.Phase
  -> Cruise.Engine.1.Temp
     Landing.Engine.1.Temp
  -> Cruise.Engine.2.Temp
     Landing.Engine.2.Temp
     Take.Off.Engine.1.RPM
  -> Cruise.Engine.1.RPM
     Take.Off.Engine.2.RPM
  -> Cruise.Engine.2.RPM
     Take.Off.Vacuum.Pressure
     Cruise.Vacuum.Pressure
     Landing.Vacuum.Pressure
     Landing.Engine.1.RPM
     Landing.Engine.2.RPM
     Take.Off.Vertical.Speed
     Fuel.Warning
  -> Gear.Up
  -> Cruise.Wing.Flap.1
  -> Cruise.Wing.Flap.2

READING AIRCRAFT CONDITION: Cruise.Vertical.Speed\ AT TIME = 312

DEACTIVATING RULE'S TASK EXECUTION LIST: Ready for cruising
```

```
**** BEGIN TASK: no_cruis ****

The aircraft is no longer in cruise mode

**** no_cruis ****


REMOVING TASK cruise FROM TASK EXECUTION LIST

READING AIRCRAFT CONDITION: Take.Off.Wing.Flap.1\ AT TIME = 312
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.1\ AT TIME = 312
READING AIRCRAFT CONDITION: Landing.Wing.Flap.1 AT TIME = 312
READING AIRCRAFT CONDITION: Take.Off.Wing.Flap.2\ AT TIME = 312
READING AIRCRAFT CONDITION: Cruise.Wing.Flap.2\ AT TIME = 312
READING AIRCRAFT CONDITION: Landing.Wing.Flap.2 AT TIME = 312
READING AIRCRAFT CONDITION: Gear.Up\ AT TIME = 360
READING AIRCRAFT CONDITION: Gear.Down AT TIME = 364

FIRING RULE: Ready for landing

**** BEGIN TASK: land ****

The aircraft is ready for landing

**** land ****

  The following conditions were true at the time the rule fired:
     Take.Off.Fuel
     Cruise.Fuel
     Landing.Fuel
  -> Landing.Vertical.Speed
     Take.Off.Flight.Phase
     Cruise.Engine.1.Temp
  -> Landing.Engine.1.Temp
     Cruise.Engine.2.Temp
  -> Landing.Engine.2.Temp
     Take.Off.Engine.1.RPM
     Cruise.Engine.1.RPM
     Take.Off.Engine.2.RPM
     Cruise.Engine.2.RPM
     Take.Off.Vacuum.Pressure
     Cruise.Vacuum.Pressure
  -> Landing.Vacuum.Pressure
  -> Landing.Engine.1.RPM
  -> Landing.Engine.2.RPM
     Take.Off.Vertical.Speed
     Fuel.Warning
  -> Landing.Wing.Flap.1
  -> Landing.Wing.Flap.2
  -> Gear.Down
```

```
READING AIRCRAFT CONDITION: Take.Off.Fuel\ AT TIME = 412
READING AIRCRAFT CONDITION: Cruise.Fuel\ AT TIME = 412
READING AIRCRAFT CONDITION: end program AT TIME = 448

Program Ends

  The following conditions were true at program termination:
     Landing.Fuel
     Landing.Vertical.Speed
     Take.Off.Flight.Phase
     Cruise.Engine.1.Temp
     Landing.Engine.1.Temp
     Cruise.Engine.2.Temp
     Landing.Engine.2.Temp
     Take.Off.Engine.1.RPM
     Cruise.Engine.1.RPM
     Take.Off.Engine.2.RPM
     Cruise.Engine.2.RPM
     Take.Off.Vacuum.Pressure
     Cruise.Vacuum.Pressure
     Landing.Vacuum.Pressure
     Landing.Engine.1.RPM
     Landing.Engine.2.RPM
     Take.Off.Vertical.Speed
     Fuel.Warning
     Landing.Wing.Flap.1
     Landing.Wing.Flap.2
     Gear.Down
     end program
```